Master’s Thesis

Terrain Aided
Underwater Navigation using
Bayesian Statistics

Tobias Karlsson

Thesis No.: LiTH-ISY-EX-3292-2002
Linköping 2002

Supervisor: M.Sc. Björn Johansson
SAAB Bofors Underwater Systems
Lic. Rickard Karlsson
ISY, Linköpings Universitet

Examiner: Prof. Fredrik Gustafsson
ISY, Linköpings Universitet

Department of Electrical Engineering
Linköpings Universitet
SE-581 83 Linköping, Sweden
Terrängstöttad undervattensnavigering baserad på Bayesiansk statistik

Terrain Aided Underwater Navigation using Bayesian Statistics

Tobias Karlsson

For many years, terrain navigation has been successfully used in military airborne applications. Terrain navigation can essentially improve the performance of traditional inertial-based navigation. The latter is typically built around gyros and accelerometers, measuring the kinetic state changes. Although inertial-based systems benefit from their high independence, they, unfortunately, suffer from increasing error-growth due to accumulation of continuous measurement errors.

Undersea, the number of options for navigation support is fairly limited. Still, the navigation accuracy demands on autonomous underwater vehicles are increasing. For many military applications, surfacing to receive a GPS position-update is not an option. Lately, some attention has, instead, shifted towards terrain aided navigation.

One fundamental aim of this work has been to show what can be done within the field of terrain aided underwater navigation, using relatively simple means. A concept has been built around a narrow-beam altimeter, measuring the depth directly beneath the vehicle as it moves ahead. To estimate the vehicle location, based on the depth measurements, a particle filter algorithm has been implemented. A number of MATLAB simulations have given a qualitative evaluation of the chosen algorithm. In order to acquire data from actual underwater terrain, a small area of the Swedish lake, Lake Vättern has been charted. Results from simulations made on this data strongly indicate that the particle filter performs surprisingly well, also within areas containing relatively modest terrain variation.
Abstract

For many years, terrain navigation has been successfully used in military airborne applications. Terrain navigation can essentially improve the performance of traditional inertial-based navigation. The latter is typically built around gyros and accelerometers, measuring the kinetic state changes. Although inertial-based systems benefit from their high independence, they, unfortunately, suffer from increasing error-growth due to accumulation of continuous measurement errors.

Undersea, the number of options for navigation support is fairly limited. Still, the navigation accuracy demands on autonomous underwater vehicles are increasing. For many military applications, surfacing to receive a GPS position-update is not an option. Lately, some attention has, instead, shifted towards terrain aided navigation.

One fundamental aim of this work has been to show what can be done within the field of terrain aided underwater navigation, using relatively simple means. A concept has been built around a narrow-beam altimeter, measuring the depth directly beneath the vehicle as it moves ahead. To estimate the vehicle location, based on the depth measurements, a particle filter algorithm has been implemented. A number of MATLAB simulations have given a qualitative evaluation of the chosen algorithm. In order to acquire data from actual underwater terrain, a small area of the Swedish lake, Lake Vättern has been charted. Results from simulations made on this data strongly indicate that the particle filter performs surprisingly well, also within areas containing relatively modest terrain variation.
Acknowledgements

Over the past few months, I have been given the opportunity to write my master’s thesis at SAAB Bofors Underwater Systems in Motala. With this project, the main part of my M.Sc. studies at Linköping University is concluded. My personal opinion about this period is that it has been great fun and highly invigorating. The work has required theoretical depth as well as practical skills, creating a challenging and rewarding daily environment. I especially appreciate the warm and generous attitude I have experienced from everyone who has been more or less involved in this project. This attitude made me feel more as part of a team, rather than as an individual writing his master’s thesis. For this I am genuinely grateful, and I would like to take this opportunity share my gratitude.

First and foremost I would like to thank those responsible at SAAB Bofors Underwater Systems for giving me the opportunity to work with this master’s thesis project. This small group of people has, from the start and all the way throughout the entire project, continuously followed and supported the progress of my work. Therefore, a major thanks to my manager M.Sc. Per Johansson and to my supervisors M.Sc. Anna Falkenberg, Dr. Björn Hedin and M.Sc. Björn Johansson. Extra credit goes to Björn Johansson, who has been working closely, and sometimes intensively, with many of the practical issues surrounding this work. Without his help, this project would not have given even remotely as good results as now, but would definitely have taken a lot more time.

I would also like to thank my supervisor at Linköping University, Lic. Rickard Karlsson, for his continuous support and his highly valuable advice. Without his immediate identification of how to approach the theoretical aspects of terrain navigation, this thesis would definitely have taken a turn for the worse. At the same time, I would like to thank my examiner Prof. Fredrik Gustafsson, also at Linköping University, who has been monitoring this project from the start.

A special thanks goes to M.Sc. Claes Wahlberg at SAAB Bofors Underwater Systems, who has taken a particular interest in the making of this thesis. His excellent knowledge, ranging from hydro acoustics to practical issues concerning digital image processing, has considerably eased my daily work. In addition, his good sense of humour, accompanied with practical jokes, also made some days much more fun. Another special thanks to Per Pelle Österberg, who provided practical assistance during the assembling of our measurement equipment, and to Tommy Oscarsson, who patiently manoeuvred the measurement boat for several long hours, back and forth throughout the entire area of the depth soundings.

Finally, I would like to thank all of you who have helped in any extent with this thesis, whether your name has been mentioned here or not. A warm thanks goes to you all…

Linköping, January 2003
Tobias Karlsson

III
Contents

1 Introduction .................................................................................................................................... 1
  1.1 Background .......................................................................................................................... 1
  1.2 Problem Formulations .......................................................................................................... 2
  1.3 Limitations .......................................................................................................................... 3
  1.4 Thesis Outline ....................................................................................................................... 4
2 Bayesian Estimation .................................................................................................................... 5
  2.1 System Description ............................................................................................................... 6
    2.1.1 Discrete-Time Representation ....................................................................................... 6
    2.1.2 The Recursive State Model ........................................................................................... 7
    2.1.3 Recursive Prediction and Evaluation .............................................................................. 7
  2.2 Recursive Bayesian Estimation .............................................................................................. 8
    2.2.1 The State Probability Density Function ......................................................................... 8
    2.2.2 The Initial Uncertainty Density ...................................................................................... 9
    2.2.3 The Measurement Update ............................................................................................. 9
    2.2.4 The Time Update ......................................................................................................... 10
    2.2.5 The Conditional Mean-Square State Estimate .............................................................. 11
    2.2.6 The Recursive Bayesian Estimation Algorithm ............................................................ 11
  2.3 The Particle Filter ................................................................................................................ 14
    2.3.1 Parallel Recursive Prediction and Evaluation ............................................................... 14
    2.3.2 Monte Carlo Integration ............................................................................................... 14
    2.3.3 Importance Sampling .................................................................................................... 15
    2.3.4 Sampling Importance Resampling (SIR) ....................................................................... 17
    2.3.5 The SIR Algorithm ....................................................................................................... 19
    2.3.6 Algorithm Divergence ................................................................................................. 19
  2.A Basic Probability Theory ..................................................................................................... 22
    2.A.1 Basic Definitions .......................................................................................................... 22
    2.A.2 Basic Notations ............................................................................................................. 22
    2.A.3 Bayes’ Theorem ........................................................................................................... 23
    2.A.4 Hidden Markov Process ............................................................................................... 23
  2.B The Update Expressions ....................................................................................................... 24
    2.B.1 The Time Update .......................................................................................................... 24
    2.B.2 The Measurement Update ............................................................................................. 25
3 Underwater Terrain Navigation ................................................................................................. 27
  3.1 Traditional Navigation Systems ............................................................................................ 27
  3.2 The Conceptual Terrain Navigation System ......................................................................... 28
  3.3 Navigation Requirements on Underwater Terrain ............................................................... 29
  3.4 Practical Aspects .................................................................................................................. 29
    3.4.1 Cost per Payload ........................................................................................................... 30
    3.4.2 Sonar Beam Refraction ................................................................................................ 30
  3.5 Single Beam Terrain Navigation ........................................................................................... 31
    3.5.1 Depth Notations .......................................................................................................... 31
  3.6 Recursive Bayesian Terrain Navigation ............................................................................... 33
    3.6.1 The TNS State Model ................................................................................................... 33
    3.6.2 The TNS Particle Filter ............................................................................................... 35
Abbreviations

AUV  Autonomous Underwater Vehicle
DGPS  Differential GPS
EKF  Extended Kalman Filter
GPS  Global Positioning System
GPSU  GPS Utility
i.i.d.  Independent Identically Distributed
IS  Importance Sampling
KF  Kalman Filter
IMU  Inertial Measurement Unit
INS  Inertial Navigation System
MC  Monte Carlo
MS  Mean Square
MTTTY  Multi-threaded TTY
pdf  Probability Density Function
PF  Particle Filter
PMF  Point Mass Filter
rms  Root Mean Square
SIR  Sampling Importance Resampling
TNS  Terrain Navigation System
UUV  Unmanned Underwater Vehicles

Notations

e_t  Residual / measurement noise at time \( t \)
\( \varepsilon_t \)  Difference between measured depth, travelling depth and terrain depth
\( f(\cdot) \)  State transition equation
\( h(x_t) \)  Expected measurement for the state \( x_t \)
\( N \)  Number of particles or samples
\( p(a) \)  pdf for the stochastic variable \( a \)
\( p(a | b) \)  pdf for the stochastic variable \( a \) given the stochastic variable \( b \)
\( p(a,b) \)  Joint pdf for the stochastic variables \( a \) and \( b \)
Pr(\cdot)  Probability
\( p_e(\cdot) \)  Measurement noise pdf
\( p_v(\cdot) \)  Process noise pdf
\( \mathbb{R}^n \)  Euclidean n-dimensional space
\( \sigma \)  Standard deviation
\( T \)  Sample time
\( \nu_t \)  Process noise at time \( t \)
\( w_{ti} \)  Importance weight / particle weight \( i \) of the state vector at time \( t \)
\( x_t \)  State vector at time \( t \)
\( x_{ti} \)  Sample \( i \) of the state vector at time \( t \)
\( y_t \)  Measurement at time \( t \)
\( \mathcal{Y}_t \)  The cumulative set of measurements up to and including time \( t \)
1 Introduction

Unmanned Underwater Vehicles (UUV) sent on extended missions have with them high demands on navigation accuracy. Traditional navigation systems, based on inertial navigation and velocity measurements, are sufficiently accurate for shorter missions. However, when the mission-duration is increased, the imperfections of a traditional navigation system begin to severely impact on the navigation results. Therefore, the possibility of supporting traditional navigation systems with position estimates based on information from the terrain beneath the vehicle will be investigated.

1.1 Background

Finding ones way, with an assuring confidence of ones current position, is a vital aim for all safe sea travel and air travel. Over the years, a variety of constantly improved positioning and navigation systems have been taken into practise, both for military and civilian use. One of the major contributions to reliable positioning was the introduction of the satellite based Global Positioning System (GPS). After the removal of the deliberately introduced positioning error, most handheld GPS receivers now have accuracy below ten meters. With the introduction of Differential GPS\(^1\), reliable positioning within one meter is often no longer any challenge. Overall, GPS receivers have evolved from being available for military use only, to become anyone's property.

Based on the GPS, there has been a rapid development of navigation, positioning and collision avoidance systems for both marine and airborne commercial vehicles. One of these systems is the aspiring international standard for GPS transponder systems, created by the Swedish inventor Håkan Lans. Even though civilian aviation may benefit greatly from these types of applications, they are of little or no use for military underwater purposes.

Many military applications cannot be allowed to rely on outside systems for their navigation. Instead they have to be autonomous and able to conduct their operations without revealing their presence. This is especially crucial for some underwater applications, which cannot receive the GPS signal in submerged positions. Due to technical difficulties, and the apparent danger of detection, surfacing to receive a position-update is generally not an option. Instead, autonomous subsurface applications have to rely on dead-reckoning based navigation systems. These are mainly based on information given by gyros, accelerometers and velocity measurements from Doppler logs or indirectly via the propulsion system. One of the main benefits of these systems is their high independence and, with exception of the Doppler log, their low risk of detection due to their passive nature. One of their main disadvantages is their time increasing positioning uncertainty due to continuous reckoning errors. Even small measurement-errors will over time give rise to relatively large accumulated positioning

\(^1\text{DGPS - Differential GPS is a GPS device, supported with an error-correction received as a RDS radio signal.}\)
errors (Figure 1.1). Also for the more accurate Inertial Navigation Systems (INS), this is a significant problem.

One solution to handle the increasing positioning inaccuracy, due to the INS drift, is to recalibrate the system against known positions before the uncertainty grows unacceptably large. Pilots have used churches and other landmarks for manual fixed-point calibration over a long time and submarine crews calculate their positioning error by comparing single sonar image of the seabed to high-resolution charts.

Another solution to reduce the positioning drift would be to continuously support the INS with information from additional sensors. This can be done using terrain navigation and has significantly improved the positioning accuracy in many airborne applications. Different approaches have been made, but generally these systems are based on real time, on-line comparison between a radar-measured terrain profile and a digital terrain database. Because of the successful use in airborne applications, terrain aided navigation is now also gaining increasing attention for underwater applications.

1.2 Problem Formulations

The intention behind this project has been to investigate how well a relatively simple terrain navigation system can perform. Previously, several approaches using 3D sonars with moving beams or multiple fixed beams have been made, e.g. [And 00] and [Nyg 99]. This, however, is not the case here. Instead, this system is based on terrain profiling using a fixed single narrow-beam sonar, placed beneath the vehicle and aimed straight at the bottom (Figure 1.2). This sonar continuously measures the terrain profile as the vehicle moves ahead, and compares it to a terrain database.
The three major objectives of this project can be specified as:

1. To give a theoretical and mathematical introduction to the terrain navigation problem, based on the use of a fixed single narrow-beam sonar as described above.
2. To create an actual terrain map over a small area of Lake Vättern (approximately 300-by-300 meters).
3. To evaluate one or more terrain navigation algorithms against both simulated and experimental terrain data, e.g. the created terrain map.

1.3 Limitations

No aspects of how to actually use the position estimate derived from the Terrain Navigation System (TNS) for vehicle guidance will be described here. Nor will any of the practical aspects of integrating a TNS into any existing or future vehicle be taken into account. The study will also be limited to examining the potential of a system based on a fixed single narrow-beam sonar, and will thus not cover any other approaches to any greater extent.

Another limitation goes to the purpose of the MATLAB simulations. The intention behind these simulations is first and foremost to give a qualitative understanding of the nature of the particle filter. The aim is also to give a general overview of what navigation results are achievable. Some general comparisons between the charted terrain and terrain containing more variation has been made. Still, the aim has not been to evaluate statistic average performance of the particle filter or how to chose the particle filter parameters in an optimal way. Therefore, no extensive Monte Carlo simulations have been made.
1.4 Thesis Outline

Chapter 1 Introduction gives the background as well as problem formulation and limitations. It also contains this thesis outline.

Chapter 2 Bayesian Estimation gives a description of the theoretical framework, the Bayesian estimation using particle filters, which is used for position estimation.

Chapter 3 Underwater Terrain Navigation gives a more detailed description of the conceptual AUV model. It also applies the theory from Chapter 2 on the terrain navigation problem.

Chapter 4 Depth Charting and Map Generation describes the depth charting procedure, the equipment used and the measurement data processing made in order to create the resulting terrain map.

Chapter 5 Simulations describes some of the qualitative simulations made in order to evaluate the particle filter performance when navigating e.g. in the created terrain map.

Chapter 6 Results summarises this work with the derived conclusions. It also contains some recommendations for future work.
2 Bayesian Estimation

This chapter gives an introduction to some fundamental aspects of recursive Bayesian estimation, and eventually also to the particle filter (PF). The need for solving a problem using particle filters primarily occurs when handling non-linear problems, and mainly those that are difficult to transform into linear ones. The terrain-navigation problem is such a case. As shown by e.g. [Ber 99], standard Bayesian approaches using extended Kalman filters (EKF) are not the optimal choice for this kind of problem, but the PF is.

The first section of this chapter gives a description of which systems the Bayesian estimation framework applies to. Then follows an introduction to the Bayesian framework. In the final section, the last steps towards the particle filter are described, presenting the Sampling Importance Resampling (SIR) algorithm. This is the particular algorithm that will be adapted to the terrain navigation problem in Chapter 3. At the end of this chapter, an appendix is given. It contains some basic probability theory and some additional derivations.

Generally, Bayesian estimation is used when estimating parameters or states of a system using noisy and indirect measurements. In the special case of Gaussian noise and linear systems, the Kalman filter (KF) gives the optimal solution to the inference problem. The KF may be adapted to also handle non-linear problems sub-optimally, using the EKF. The condition, though, is Gaussian noise and that the problem can be linearised locally around the parameters of interest. However, if the system is affected by non-Gaussian noise, the KF or the EKF will not always perform optimally. Instead, algorithms like the PF or the point mass filter (PMF) gives the optimal solution to the non-linear, non-Gaussian inference problems. Though the PF and the PMF are both based on the Bayesian framework described here, only the PF implementation will be given.

A general system, to which the Bayesian estimation applies, can be described by

\[ x_{t+1} = f_t(x_t, v_t) \]  \hspace{1cm} (2.1a)
\[ y_t = h(x_t) + e_t \]  \hspace{1cm} (2.1b)

where \( x \) is the state of interest and \( y \) is the indirect measurement, modelled as a function of the state and affected by a measurement noise \( e \). The time dependent evolution of \( x \) is modelled by the function \( f \), which also takes into account the uncertainty of the exact value of \( x \), modelled by the process noise \( v \). The process noise and the measurement noise are described by their distributions \( p_v \) and \( p_e \) respectively. (2.1a) and (2.1b) are referred to as the state equation and the measurement equation.

In short, the Bayesian framework used for recursively describing the system above, is given by the equations
\[
p(x_{t+1} \mid y_t) = \int_{\mathbb{R}} p(x_{t+1} \mid x_t)p(x_t \mid y_t)dx_t \\
p(x_t \mid y_t) = \frac{p(y_t \mid x_t)p(x_t \mid \mathcal{Y}_{t-1})}{p(y_t \mid \mathcal{Y}_{t-1})}
\]

These equations are referred to as the time update and the measurement update respectively, and describe the probability of any given state \(x\), given all previous observations. The main part of this chapter will be used to describe the origin and meaning of (2.2a) and (2.2b) more in detail.

For more comprehensive descriptions of other approaches to the terrain navigation problem, e.g. the already mentioned EKF and PMF, the reader is referred to for instance [And 79]. In addition, [Ber 99] together with [Dou 98] presents most of the definitions and notations of the Bayesian framework used here.

### 2.1 System Description

The need to estimate the state \(x\) of any system mainly occurs if the quantity of interest cannot be directly measured. Instead, a state-estimate \(\hat{x}\) has to be created using an indirect observation \(y\). In the case of terrain navigation, the state variable \(x\) can be chosen to represent the true position, while \(y\) provides information from the observed terrain. Naturally, the observation \(y\) and the true position \(x\) are somehow related. If the system applies to the Bayesian framework, the recursive state model gives a general description of such a relation.

#### 2.1.1 Discrete-Time Representation

Even though the actual process described by \(x\) in many cases is a time-continuous process \(x(\tau)\) (for instance if \(x\) describes some physical quantity such as speed, position or altitude) the most convenient way to represent \(x\) would be to use a time-discrete model. This is even more natural, as many measurements are not continuously received (or, at least, not continuously stored for computational use), but instead sampled at discrete times separated by a sample time \(T\). Here, this dependence on \(T\) will usually not be explicitly stated. Instead, one specific value \(x(\tau) = x(t \cdot T)\) will be written as \(x_t\). Thus, the sub-indices indicates that \(x_t\) and \(x_{t+1}\) are separated by one sample time \(T\), starting at time \(\tau = t \cdot T\). Consequently, \(x(0)\) is written as \(x_0\) and represents the value of \(x\) given at time \(\tau = 0\). The same principle of sub-indices applies to most parameters, variables or functions here.
2.1.2 The Recursive State Model

A general model describing the time-dependent recursive relationships of an evolving hidden Markov process\(^2\) \(x\) and its corresponding conditionally independent observation \(y\) is given by

\[
\begin{align*}
    x_{t+1} &= f_t(x_t, v_t) \tag{2.3a} \\
    y_t &= h(x_t) + e_t \tag{2.3b}
\end{align*}
\]

\(x \in \mathbb{R}^n\) and \(y \in \mathbb{R}^m\).

These two equations are called the transition equation or the state equation and the observation equation or the measurement equation respectively. Here, the general non-linear function \(f_t(x_t, v_t)\) describes the state transition from time \(t\) to time \(t+1\). The time-invariant function \(h(\cdot)\) describes the modelled (or perhaps even the true) value of the expected observation \(y\) for any given state \(x\). Note that the function \(f_t\) can vary over time, as indicated by its time index \(t\). Also note that there is no distinction made between scalars or vectors for \(x\) and \(y\). The state variable \(x\) is simply assumed to belong to some \(n\)-dimensional space and the observation \(y\) is similarly assumed to belong to some \(m\)-dimensional space.

The state process \(x\) and the observation \(y\) can be seen as stochastic variables. Due to this stochastic nature, the transition equation contains a noise component \(v_t\). So does the measurement equation with its noise component \(e_t\), also called the residual. Both of these noise components are introduced mainly due to inaccuracies in \(f_t\) and \(h(\cdot)\) respectively. They also reflect outside disturbances affecting the state variables \(x\) (system noise / process noise) or inaccurate measurements giving false values of \(y\) (measurement noise). The nature of both \(v_t\) and \(e_t\) are generally considered to be known for each specific system, and are hence described by their probability density functions \(p_{v_t}(\cdot)\) and \(p_{e_t}(\cdot)\) respectively.

2.1.3 Recursive Prediction and Evaluation

The transition equation (2.3a) could be used to predict the next state \(x_{t+1}\), given the current state \(x_t\). If the initial true state is given by \(x_0\), a strait forward \(n\)-step prediction could be made applying the transition equation \(n\) consecutive time on the previous result, starting with \(x_0\). Due to the stochastic nature of \(f_t\), given by \(v_t\), the resulting \(n\)-step prediction would not yield a deterministic result, but instead result in different values for each new attempt. Thus, a simple \(n\)-step prediction would not give a very accurate estimate of \(x\). To make things even worse, the true starting value \(x_0\) is many times just vaguely known due to the hidden nature of \(x\).

A straightforward \(n\)-step prediction would neither take into account the information given by the observation \(y_t\) when predicting one specific value \(x_{t+1}\). The residual \(e_t\),

\(^2\) For the “hidden Markov” property, see Appendix 2.A.4.
describing the difference between the observed value $y_t$ and the anticipated, modelled value $h(x_t)$, obviously gives a quality-measure of any $n$-step prediction $x_t$. A large residual indicates a false prediction, while a small residual indicates a good possibility of a true prediction. This could be used to evaluate the probability of each predicted state value. Given that the noise distribution $p_{ei}(\cdot)$ of the residual is known, the residual can be used to calculate the probability of receiving an observation $y_t$, given the current state $x_t$, as illustrated by

$$e_t = y_t - h(x_t) \implies p_{ei}(e_t) = p_{ei}(y_t - h(x_t)).$$

(2.4)

Thus, the likelihood $p(y_t | x_t)$ would be proportional to the residual given by the measurement equation (2.3b), as indicated by

$$p(y_t | x_t) \propto p_{ei}(e_t).$$

(2.5)

Still, the predicted value from the transition equation, combined with a quality evaluation from the measurement equation, would not provide a satisfactory state-estimate. However, these principals are the foundation for more complex algorithms.

2.2 Recursive Bayesian Estimation

To give a comprehensive description of the hidden Markov process $x$, the Bayesian approach uses an indirect description, namely the probability density function (pdf). This function naturally describes the probability for $x$ assuming each specific value of all possible ones. The pdf is constantly changing shape, either due to outside signals (such as vehicle movement in the terrain navigation case) or due to receiving indirect information from new measurements $y$. This is the reason for making the state model a recursive one. Thus, the pdf will evolve over time, as it is constantly updated to give the best possible description of $x$, based on all previous observations.

2.2.1 The State Probability Density Function

One natural way of describing the probability of receiving one particular outcome of a test-run would be to use a pdf. A simulation, starting with a given state $x_0$ at time $\tau = 0$ and ending with the estimated state $x_t$ at time $\tau = t \cdot T$, could be described by its resulting estimated consecutive set of state-values $\{x_0, x_1, \ldots, x_t\}$, provided its corresponding set of observations $\{y_0, y_1, \ldots, y_t\}$. This set of observations is denoted as

$$\mathcal{Y}_t = \{y_i\}_{i=0}^t.$$

(2.6)

Thus, all the relevant information about the probability of any given test-run would be contained in the pdf $p(x_0, x_1, \ldots, x_t | y_0, y_1, \ldots, y_t)$. If such a continuous description should exist, the probability of any given combination of consecutive states and corresponding measurements could be described.
When using recursive Bayesian estimation, the knowledge about the probability of an entire test-run is not needed to predict the next state $x_{t+1}$. The reason for this is the Markovian property $p(x_{t+1} \mid x_t) = p(x_t \mid x_0, x_1, \ldots, x_0)$ of the state process $x$. Instead, the momentary state pdf $p(x_t \mid \mathcal{Y}_t)$ is enough to provide a recursive description of the current most probable value of $x$.

### 2.2.2 The Initial Uncertainty Density

In order to recursively estimate the pdf of $x$, an initial pdf is needed. Due to the unobservable nature of $x$, the true starting-state is, many times, not entirely known. Instead, the user of the algorithm has to provide an approximate initial state $x_0$, along with some accuracy assessment of that approximation. The most suitable accuracy assessment would be a pdf $p(x_0)$. In a terrain navigation application, $x_0$ could be the believed (or perhaps even the true) starting position. Consequently, $p(x_0)$ would then describe the probability that the user-provided initial state $x_0$ is the true starting-state, based on the positioning accuracy of $x_0$. Generally, the distribution corresponding to $p(x_0)$ could be of almost any kind, such as Gaussian, uniform or some problem-specific user-defined distribution, depending on how accurately $x_0$ is determined. Many times, $p(x_0)$ is referred to as the initial uncertainty density.

### 2.2.3 The Measurement Update

In order to give the most accurate description of $x$, the pdf $p(x_t)$ would have to be reshaped after receiving each new observation $y_t$. This is done recursively, using the measurement update equation given by

$$
p(x_t \mid \mathcal{Y}_t) = \frac{p(y_t \mid x_t) p(x_t \mid \mathcal{Y}_{t-1})}{p(y_t \mid \mathcal{Y}_{t-1})}.
$$

The result from the measurement update is the posterior density $p(x_t \mid \mathcal{Y}_t)$, i.e. the probability density function posterior to the measurement update. As seen above, the measurement update consists of three different factors. Firstly the likelihood $p(y_t \mid x_t)$, secondly the prior $p(x_t \mid \mathcal{Y}_{t-1})$ and thirdly the total probability of $y_t$ given by $p(y_t \mid \mathcal{Y}_{t-1})$. The origin of these densities and their importance for the posterior density are somewhat different.

The prior density $p(x_t \mid \mathcal{Y}_{t-1})$ naturally corresponds to the probability of $x_t$ being the true state, given that $\mathcal{Y}_{t-1}$ is the received set of observations. The naming of the prior density comes from its relation to the measurement $y_t$, i.e. $p(x_t \mid \mathcal{Y}_{t-1})$ is the probability density function prior to the measurement update. In other words, the prior describes $x$ just before a new measurement is received.

The likelihood $p(y_t \mid x_t)$ is indirectly given by the residual $e_t$ from the measurement equation (2.3b). If $p_{e_t}(\cdot)$ is known, the likelihood of receiving the measurement $y_t$ could...
be calculated for any given state \( x_t \), using basic probability theory. As indicated by the time indices, the likelihood is calculated after a new measurement is received.

The third factor of (2.7) is the total probability of \( y_t \), given by \( p(y_t \mid \mathcal{Y}_{t+1}) \). However, when calculating the posterior density, the observation \( y_t \) is already at hand. Therefore, \( p(y_t \mid \mathcal{Y}_{t+1}) \) can be regarded merely as a normalising factor. Consequently, it is not necessary to calculate the actual value of \( p(y_t \mid \mathcal{Y}_{t+1}) \).

At the start-up of the algorithm, \( p(x_t \mid \mathcal{Y}_{t-1}) \) is equal to the initial uncertainty-density \( p(x_0 \mid \mathcal{Y}_{-1}) = p(x_0) \). Here, \( \mathcal{Y}_{-1} \) is the set of observations at time \( \tau = -T \) denoted according to (2.6). Since there are no observations available before the time \( t = 0 \), \( \mathcal{Y}_{-1} \) is consequently an empty set. This notation should be seen merely as a means to ensure an unambiguous presentation. Starting with the uncertainty-density \( p(x_0) \), the recursion begins when the first observation \( y_0 \) is received. This observation is used when calculating the likelihood and applying the measurement update (2.7) to the current pdf.

As mentioned before, the exact value of the normalising factor \( p(y_t \mid \mathcal{Y}_{t-1}) \) is not of any interest in this case. A state-estimate, for instance based on the peak-value of the pdf, could still be calculated without exact normalisation. This benefit is used by the SIR algorithm, described in Section 2.3.4 (though is does not use peak-value estimates), which for numerical reasons instead normalises the pdf to integrate to unity each time it is calculated. Consequently, the posterior density is given accurately enough by the simplified version of (2.7) as

\[
p(x_t \mid \mathcal{Y}_t) \propto p(y_t \mid x_t) p(x_t \mid \mathcal{Y}_{t-1}),
\]

(2.8)

In conclusion: When receiving a new measurement, the likelihood is calculated using the measurement equation, while the prior is already given by the previous step of the iterative process. This gives the non-normalised posterior density according to (2.8). More information on \( p(y_t \mid \mathcal{Y}_{t+1}) \) and the deriving of (2.7) is given in Appendix 2.B.

### 2.2.4 The Time Update

The values of the hidden Markov process \( x \) will constantly evolve over time. This propagation is described by the transition equation (2.3a), and is reflected by the shape of the pdf used to describe \( x \). The reshaping of the pdf is done using the **time update** equation

\[
p(x_{t+1} \mid \mathcal{Y}_t) = \int_{\mathbb{R}^n} p(x_{t+1} \mid x_t) p(x_t \mid \mathcal{Y}_t) dx_t.
\]

(2.9)

After calculating the posterior \( p(x_t \mid \mathcal{Y}_t) \) (or at least calculating it up to a normalising factor) this is the final step to complete one cycle of the recursive process of estimating \( x \). At this stage, the shape of the pdf \( p(x_{t+1} \mid \mathcal{Y}_t) \) of the following state \( x_{t+1} \) is predicted, based on the received observations so far.
When the time update is applied, the posterior pdf, normalised or not, is recursively known from the measurement update given by (2.7) or (2.8). In order to propagate the pdf, i.e. to predict how the pdf would appear at the next sampling moment, the transition pdf \( p(x_{t+1} \mid x_t) \) must be known. Here, it is given by the transition equation (2.3a). Provided that the noise distribution \( p_v(\cdot) \) is known, the transition pdf simply describes the modelled (or perhaps even the true) behaviour of the state process as it evolves between two sampling moments.

When the time update has been used to calculate \( p(x_{t+1} \mid \mathcal{Y}_t) \), one cycle of the recursive probability density estimation is completed. Then, the time index is increased one step from \( t \) to \( t+1 \), and the iterative cycle starts over with \( p(x_{t+1} \mid \mathcal{Y}_t) \) as the new prior density \( p(x_t \mid \mathcal{Y}_{t-1}) \). This way, a pdf that estimates the most likely values of the hidden Markov process \( x \), based on the modelled behaviour of \( x \) and the received observations \( y \), is recursively propagated and reshaped. More information on the deriving of (2.9) is given in Appendix 2.B.

### 2.2.5 The Conditional Mean-Square State Estimate

Though the pdf provides a comprehensive and general solution to the inference problem of estimating \( x \), it gives a quite complex description of the state estimate \( \hat{x} \). As suggested in previous sections, a maximum peak-value estimate could be used instead. An even better estimate would be based on the expectation value of the pdf. As could be recalled from most elementary courses in probability theory, the general expression for the expectation value of a variable \( z \), provided its pdf \( p(z) \), is given by

\[
E\{z\} = \int_{\mathbb{R}^p} z \cdot p(z) dz, \quad z \in \mathbb{R}^p.
\]  

One state estimate, closely related to the expectation value of \( E\{x\} \), is given by the conditional mean-square estimate

\[
\hat{x}_{t}^{MS} = \int_{\mathbb{R}^p} x_i \cdot p(\mathcal{Y}_t \mid x_i) dx_i.
\]  

This is the optimal estimate seen from a mean-square point of view, as shown in [Ber99, p. 24].

### 2.2.6 The Recursive Bayesian Estimation Algorithm

To summarise, the recursive Bayesian estimation is described by the initial uncertainty-density, the measurement update and the time update. The entire recursive algorithm can be described in the following steps:

0. **Initialisation.** The Bayesian estimation algorithm is initialised at the time index \( t = 0 \) by a user provided initial uncertainty-density \( p(x_0) = p(x_0 \mid \mathcal{Y}_{-1}) \), describing the probability of \( x_0 \) being the true starting state.
1. **Measurement Update.** When a new observation $y_t$ is received, the prior $p(x_t \mid Y_{t-1})$ is updated using the measurement update, resulting in the posterior $p(x_t \mid Y_t)$.

2. **Time Update:** The posterior is predicted one step ahead in time using the time update, resulting in $p(x_{t+1} \mid Y_t)$.

3. **Time Increase:** The time index is increased from $t$ to $t+1$ and the algorithm iterates to item 1, with the predicted pdf $p(x_{t+1} \mid Y_t)$ from item 2 as the new prior $p(x_t \mid Y_{t-1})$.

To conclude the description of the recursive state estimation, the calculation of the state estimate (2.11) could be inserted between item 1 and 2. The general behaviour of the Bayesian estimation algorithm is also described by Figure 2.1 to 2.3. What now remains is to implement this algorithm in a way that does not require continuous descriptions. One solution to this is the particle filter.
Figure 2.1  The Recursive Bayesian Estimation Algorithm is initialised by a user provided initial uncertainty density $p(x_0) = p(x_0 \mid Y_{-1})$, describing the most probable location of the starting state $x_0$.

Figure 2.2  When a new observation $y_t$ is received, the prior $p(x_t \mid Y_{t-1})$ is resized, eliminating the least likely positions. The result is the posterior $p(x_t \mid Y_t)$.

Figure 2.3  The posterior is predicted one step ahead, resulting in $p(x_{t+1} \mid Y_t)$, which will become the new prior $p(x_t \mid Y_{t-1})$ when $t$ is increased. The algorithm will then use the next measurement to reshape this pdf into a new posterior $p(x_t \mid Y_t)$, and so on...
2.3 The Particle Filter

The particle filter (PF) is a discrete implementation of Bayesian estimation using Monte Carlo integration. Following the space-continuous Bayesian framework, the particle filter recursively propagates and reshapes a pdf describing what is currently known about the system. However, it differs from the general framework in the way that it does not use a complete analytical/continuous description of the pdf. Instead, the PF uses a discrete representation of the pdf called a particle cloud. One specific PF implementation, the Sampling Importance Resampling (SIR) algorithm or the Bayesian Bootstrap algorithm, is described at the end of this subchapter.

This section starts with a schematic description of the simulation-based principles behind the PF. After that follows an introduction to the Monte Carlo integration, using Riemann-sums to approximate continuous functions with discrete representations. Then, the Monte Carlo integration is applied to the Bayesian estimation in the section about Importance Sampling (IS), followed by two sections about the SIR algorithm. These two sections describe the principles of the PF implementation that will be used during the simulations in Chapter 5. Finally, some comments on the risk of algorithm divergence and the means to detect such an event are given.

2.3.1 Parallel Recursive Prediction and Evaluation

In order to find a more accurate state estimate than a single $n$-step predicted state value (as suggested in Section 2.1.3) a more complex method is required. One way to accomplish this would be to use some sort of simulation-based method. Assuming that the starting-state $x_0$ is relatively well known, a large number of parallel $n$-step predictions could be made. One by one, these $n$-step predictions, called test-runs or simulations, would only provide one example of a possible outcome of the state variable, provided the initial state $x_0$. Together, though, a large number of different simulations, based on the same initial state and evaluated against the residual $e_r$, would provide a relatively good recursive knowledge about the possible true states. This is basically what is done by the PF. Here, several parallel test-runs are made. Test-runs with too large residuals are terminated after some time, while test-runs with small residuals are duplicated. This is done in order for the PF algorithm to more thoroughly explore the possible outcome of that particular starting-state.

2.3.2 Monte Carlo Integration

When implementing the Bayesian estimation framework in a non-continuous way, the calculations on the underlying pdf:s can not be made analytically. Instead, numerical methods such as Monte Carlo integration must be used. Using stochastic Riemann-sum approximations, Monte Carlo integration avoids explicit analytic expressions. Instead, it uses a quantification of the state space over which the integrals are evaluated.
Monte Carlo integration starts with the general expression for analytically solving an integral $I$ of a function $\varphi(x)$ over a specified domain $D \subseteq \mathbb{R}^n$, written as

$$I = \int_{\mathbb{R}^n} \varphi(x) dx. \quad (2.12)$$

If $\varphi(x)$ can be factorised as $\varphi(x) = \varphi(x) / g(x) \cdot g(x) = \pi(x) \cdot g(x)$, where $\pi(x) = \varphi(x) / g(x)$ is positive and integrates to unity

$$\int_{\mathbb{R}^n} \pi(x) dx = 1, \quad \pi(x) \geq 0, x \in \mathbb{R}^n, \quad (2.13)$$

then (2.12) can be rewritten according to

$$I = \int_{\mathbb{R}^n} g(x) \pi(x) dx. \quad (2.14)$$

Due to its properties of being positive and integrating to unity, $\pi(x)$ can be regarded as a pdf describing the probability of the function $g(\cdot)$ assuming one specific value $g(x)$. Thus, the integral $I$ is actually the expectation value $E\{g(x)\}$, as follows by

$$I = \int_{\mathbb{R}^n} g(x) \pi(x) dx \equiv E\{g(x)\}. \quad (2.15)$$

If it is possible to draw $N >> 1$ independent identically distributed samples $\{x^i\}_{i=1}^N$ from $\pi(x)$, the integral $I$ can be approximated by a stochastic Riemann-sum approximation, given by

$$I = \int_{\mathbb{R}^n} g(x) \pi(x) dx \approx \frac{1}{N} \sum_{i=1}^N g(x^i). \quad (2.16)$$

If $N$ is sufficiently large, the approximation (2.16) will converge towards the true value according to the strong law of large numbers. Consequently, Monte Carlo integration can be used to give an approximation of the expectation value of the function $g(x)$. For deeper analysis of stochastic Riemann-sum approximation and its convergence, see i.e. [Ber 99, p.103].

2.3.3 Importance Sampling

Unfortunately, the nature of $\pi(x)$ in Section 2.3.2 is often not entirely known. This is handled by the Importance Sampling (IS), where $\pi(x)$ only needs to be known up to a normalising factor. Instead of drawing the samples directly from $\pi(x)$, $N >> 1$ independent identically distributed samples $\{x^i\}_{i=1}^N$ are drawn from an importance function $q(x)$. The only assumption made on $q(x)$ is that its support set covers the support set of $\pi(x)$, i.e. that $\pi(x) > 0 \Rightarrow q(x) > 0$ for all $x \in \mathbb{R}^n$. If this is the case, (2.14) can be rewritten as
\[ I = E\{g(x)\} = \int_{\mathbb{R}} g(x)\pi(x)dx = \int_{\mathbb{R}} g(x)\frac{\pi(x)}{q(x)}q(x)dx. \] (2.17)

The drawn set of samples from \( q(x) \) can now be used to create a Monte Carlo estimate of \( I \), creating a weighted sum \( g_N \) given by

\[ g_N = \frac{1}{N} \sum_{i=1}^{N} g(x^i)w^i, \quad \text{where} \quad w^i = \frac{\pi(x^i)}{q(x^i)}. \] (2.18)

The parameters \( w^i = w(x^i) \) are called the **importance weights**. If the scale factor between \( \pi(x) \) and \( q(x) \) is unknown, \( w(x) \) can only be calculated up to a normalising factor, as mentioned before. However, normalisation may be performed afterwards, given by

\[ g_N = \frac{1}{N} \sum_{i=1}^{N} g(x^i)w(x^i), \quad \text{where} \quad w(x^i) \propto \frac{\pi(x^i)}{q(x^i)}. \] (2.19)

It can be shown (see [Ber 99, p.108] and references given there) that (2.19) converges most often, or that

\[ \Pr\left( \lim_{N \to \infty} g_N = I \right) = 1. \] (2.20)

Consequently, the resulting estimate will be asymptotically unbiased for large \( N \).

When the IS method is applied to the Bayesian estimation, \( \pi(x) \) is chosen as

\[ \pi(x) = p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} \propto p(y \mid x)p(x). \] (2.21)

Here, the prior \( p(x) \) makes a satisfactory importance function \( q(x) \). If (2.17) and (2.18) was strictly followed, the importance weights would be given by

\[ w^i = \frac{p(y^i \mid x^i)}{p(y^i)}. \] (2.22)

This, however, is not the case here. Instead, the importance weights are chosen as

\[ w^i = p(y^i \mid x^i), \] (2.23)

ignoring the scale-factor \( p(y) \). The relative shape of the pdf approximation is still valid though, even if each specific value of the approximation would need to be re-scaled with \( p(y) \) to give the true pdf approximation. When calculating the expectation value of
17

$x$, the relative shape of the pdf is all that is needed, and consequently $p(y)$ can be ignored for now. Hence, it is possible to make an approximation $x_N$ of the state estimate $\hat{x}$ using (2.18) as

$$
x_N = \sum_{i=1}^{N} x^i \cdot p(y^i \mid x^i).
$$

(2.24)

When comparing (2.24) to the description given above, the estimated function $g(x)$ is easily identified as $x$ itself. The stochastic Riemann-sum approximation is taken over a set $\{x^i\}_{i=1}^N$ sampled from $p(x)$, which serves as the importance function. In an actual implementation, the importance weights would be normalised to summarise to unity. This would handle any potential numerical problem sprung from the absence of normalisation within a possibly recursive algorithm.

### 2.3.4 Sampling Importance Resampling (SIR)

When the IS method is applied to the recursive Bayesian estimation algorithm from Section 2.2, almost all components of the particle filter have been presented. What is still lacking is a resampling step, which is the way the measurement update is implemented in the discrete representation. This step is introduced here with the Sampling Importance Resampling (SIR) algorithm.

Like the IS method, the SIR algorithm starts with an approximate draw of $N>>1$ samples $\{x^i\}_{i=1}^N$ from the prior density $p(x_t \mid x_{t-1})$. These samples are denoted particles. Together, the particles form a particle cloud, which gives a good approximation of the prior as

$$
p(x_t \mid x_{t-1}) \approx \sum_{i=1}^{N} w^i \cdot \delta(x_t - x^i).
$$

(2.25)

As with the IS method, the importance weights $w^i$ are given by (2.23). The particles and their importance weight, corresponding to the shape of the approximated pdf, are illustrated by Figure 2.4.

**Figure 2.4** The particle cloud and their corresponding importance weight give a good description of the approximated pdf.
After receiving a new measurement $y_t$, the old set of particles is replaced by a new set $(x_{t+1}^i)_{i=1}^N$, which also has $N$ number of particles but instead describes the posterior $p(x_t \mid y').$ The new set is obtained by resampling with replacement from the old set. The resampling procedure is done according to the following principle:

As soon as a new measurement $y_t$ is received, each of the old particles are assigned a new importance weight $w(x_t^i) = p(y_t \mid x_t^i)$, corresponding to the likelihood given by the new measurement. When generating each of the $N$ new particles $x_{t+1}^i$, the probability of resampling (randomly picking, choosing, drawing etc.) each of the old particles $x_t^i$ is proportional to the importance weight of that particle. Each particle may be resampled one time, several times or not at all, depending on the size of its importance weight. Consequently, some particles (which had a high probability in the previous set) will be represented by several copies while others (which originally had a relatively low probability) will not be represented at all. Hence, the resampling step is the discrete version of the measurement update, transforming the old particle cloud representing the prior $p(x_t \mid y_{t-1})$ into a new one representing the posterior $p(x_t \mid y')$. For more information on how to practically implement a resampling step, see [Ber 99, p. 128].

As mentioned above, the resampled set gives a good approximation of the posterior $p(x_t \mid y')$. When applying the time update to this set of particles, the next prior will be derived. A Monte Carlo / Riemann-sum approximation of the time update equation (2.9) turns the integral into a summation, given by

$$p(x_{t+1} \mid y'_t) \approx \frac{1}{N} \sum_{i=1}^N p(x_{t+1} \mid x_{t+1}^i).$$

(2.26)

This will be the next prior, represented by the set $(x_{t+1}^i)_{i=1}^N$. The new particle cloud is obtained by applying the time update equation (2.3a) to each particle individually. This is the actual prediction step, which will make the particles explore the state space. The reason to why this propagation is not a straight forward deterministic function instead of a pdf representation, is the stochastic nature of the time update, given by $p_v(\cdot)$. This completes one recursive cycle, resulting in a particle cloud representing the next prior $p(x_{t+1} \mid y')$. After increasing $t$, the algorithm starts over with a new measurement update and so on.

What remains to complete the particle filter is a state estimate based on the derived particles. The shape of the particle cloud naturally describes the probability of finding the true state $x_t$ in any given sub-region of the state-space. The probability is proportional to the number of particles found in that region, when also considering their importance weights. How to create a suitable position estimate has already been indicated e.g. in Section 2.2.5. Here, an approximation of the conditional mean square estimate given by (2.11) becomes
\[ \hat{x}_{t}^{MS} = \int_{\mathbb{R}^d} x_{t} \cdot p(x_{t} \mid y_{t}) dx_{t} = \sum_{i=1}^{N} w_{i} \cdot x_{i}^{*}. \] (2.27)

It would also be possible to make a corresponding state estimate based on the resampled set of particles. The conditional mean square estimate is then instead given by

\[ \hat{x}_{t}^{MS} = \frac{1}{N} \sum_{i=1}^{N} x_{i}^{*}. \] (2.28)

### 2.3.5 The SIR Algorithm

The entire SIR algorithm is summarised in the following 6 items:

1. Start at \( t=0 \). Generate \( N \) samples \( \{x_{0}^{i}\}_{i=1}^{N} \) from an initial known density \( p(x_{0}) \).
2. Calculate the importance weights \( w_{i} = p(y_{t} \mid x_{i}) \) for \( i = 1, \ldots, N \) after receiving a new observation \( y_{t} \).
3. Normalise the weights \( w_{i} := \gamma^{-1} \cdot w_{i} \), where \( \gamma = \sum_{j=1}^{N} w_{j} \).
4. Generate the new set \( \{x_{t}^{*i}\}_{i=1}^{N} \) by resampling with resampling from the old set. Let the probability of resampling one specific sample be given by \( Pr(x_{t}^{*i} = x_{t}^{j}) = w_{i} \).
5. Predict each of the new samples one time, generating the set \( \{x_{t+1}^{j}\}_{j=1}^{N} \), where \( x_{t+1}^{j} \sim p(x_{t+1} \mid x_{t}^{*i}) \) for \( i = 1, \ldots, N \).
6. Increase \( t \) and continue at item 2.

The calculation of the mean square state estimate \( \hat{x}_{t}^{MS} \) can be made either between item 3 and 4 using (2.27) or between item 4 and 5 using (2.28). Besides the resampling step, the main behaviour of the particle filter is illustrated by Figure 2.5 to 2.7.

### 2.3.6 Algorithm Divergence

In every recursive estimation problem, there is always a risk that the algorithm estimate diverges to far away from the true state of the system. This will naturally become the case if the true starting state lies significantly outside the area indicated by the user-estimated initial uncertainty density. Other reasons for filter divergence could be an inaccurately modelled function \( h(\cdot) \) in the measurement equation or unexpectedly large measurement errors or process noise. Regardless of the reason, the possibility of particle filter divergence must be reckoned with.

One way to detect a possible particle filter divergence is to continuously monitor the average non-normalised importance weight of the entire set of particles. If the main part
of the particle cloud remains close to the true state, the majority of the particles should have rather high importance weights. Correspondingly, if the main part of the particle cloud is in an area of the state space far away from the true state, the majority of the particles should have relatively low importance weights.

Depending on the nature of the distribution describing the likelihood \( p(y \mid x) \), the maximum value \( A = \Pr(y - h(x) = 0) = \Pr(e = 0) \) could be determined. The value \( A \) is the highest possible value of the importance weight for one particle, and naturally corresponds to the probability of a zero residual. If all non-normalised importance weights are summarised and compared to a threshold value, a particle filter divergence could be detected. If this threshold value e.g. is chosen as \( 2/3 \) of \( A \), the sum is given by

\[
\sum_{i=1}^{N} w_i \leq \frac{2}{3} A \cdot N. \tag{2.29}
\]

When this sum, the total non-normalised importance weight, falls below the threshold value during a certain number of consecutive iterations, it could be seen as a strong indication to particle filter divergence. Note though, that isolated dips below the threshold value should not be seen as divergence, but rather as significant filter excitation from the measurements, eliminating many of the less probable particles.
Figure 2.5  The initial particle cloud given by \( \{x_0^i\}_{i=1}^N \). This set of particles is a discrete representation of the initial uncertainty density \( p(x_0) = p(x_0 \mid Y_{-1}) \).

Figure 2.6  When a new observation \( y_t \) is received, the importance-weight of each particle is changed, reshaping the representation the prior \( p(x_t \mid Y_{t-1}) \). At this point, a resampling step would normally be applied, resulting in the posterior \( p(x_{t+1} \mid Y_t) \).

Figure 2.7  Note that no resampling step has been applied to this figure. If so, many of the particles with low importance weight would not have been copied. However, each particle of the cloud has been predicted one step ahead. (Compare to Figure 2.1 to 2.3)
Appendix

2.A Basic Probability Theory

Most of the probability theory used in this chapter could be found in most literature written on the subject. Still, for the convenience of the reader, some basic definitions and notations together with the definitions of Bayes’ Theorem and the Hidden Markov property are given here.

2.A.1 Basic Definitions

A random variable \( a \) is a real-valued function whose domain is a probability space \( S \). The set \( \{ a \leq a \} \) is called an event for any real number \( a \), describing a certain subset of \( S \). An event could be said to contain a collection of outcomes, each assigned a certain probability. These probabilities are given by a measure \( Pr \), such that \( Pr(S) = 1 \). In addition, the probabilities of the events \( \{ a = +\infty \} \) and \( \{ a = -\infty \} \) must be equal to zero.

The distinction between the random variable \( a \) in general and one particular value \( a \), is in this appendix made by using bold characters. The probability \( Pr(a \leq a) \) of an event is described by the distribution function of \( a \), given by

\[
P_a(a) = Pr(a \leq a).
\] (2.30)

Based on the distribution function, the probability density function (pdf) can be defined as its derivative

\[
p_a(a) = \frac{d}{da} P_a(a).
\] (2.31)

Often the sub-indices are left out, so that \( p_a(a) \) and \( P_a(a) \) are written as \( p(a) \) and \( P(a) \), if there is no risk of ambiguity. Also note, that the labels density and distribution describing the probabilities of some specific outcome or event, sometimes are used somewhat recklessly due to their related nature.

2.A.2 Basic Notations

Given the stochastic variables \( a \) and \( b \), their pdf:s are denoted \( p_a(a) \) and \( p_b(b) \) respectively. Thus, the probability \( Pr(a = c) \) that the variable \( a \) assumes the specific value \( c \) is given by \( p_a(c) \). For notational conveniences, the indices \( a \) and \( b \) will not be explicitly written out, unless there is an apparent risk of a mix up otherwise. Instead, the pdf’s corresponding to each variable is assumed to be used, and the densities will simply be written as \( p(a) \) and \( p(b) \).
The probability $Pr(a = c, b = d)$ that $a$ and $b$ at the same time assumes the specific values $c$ and $d$ respectively, is described by the joint pdf $p_{a,b}(a,b)$. The actual probability is given by $Pr(a = c, b = d) = p_{a,b}(c,d)$. In most cases, this notation is also simplified from $p_{a,b}(a,b)$ to $p(a,b)$.

If the value of the stochastic variable $b$ is known to be $b = d$, the conditional probability $Pr(a = c \text{ given } b = d)$ of $a = c$, is described by the conditional pdf $p_{a|b}(a | b)$, often written as $p(a | b)$. Similar to the joint probability density, the actual conditional probability is given by $Pr(a = c \text{ given } b = d) = p_{a|b}(c | d)$.

2.A.3 Bayes' Theorem

Suppose that $x$ and $y$ are stochastic variables with known pdf:s $p(x)$ and $p(y)$. Furthermore, let $x$ and $y$ be scalars or vectors. The relation between the joint probability densities $p(x,y) = p(y,x)$, the conditional probability densities $p(x | y)$ and $p(y | x)$ and the single probability densities $p(x)$ and $p(y)$ is defined as

$$p(x, y) = p(x | y) p(y) = p(y | x) p(x),$$

(2.32)

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ respectively. This can be rewritten as

$$p(x | y) = \frac{p(x, y)}{p(y)} = \frac{p(y, x)}{p(y)} = \frac{p(y | x) p(x)}{p(y)},$$

(2.33)

resulting in Bayes' theorem

$$p(x | y) = \frac{p(y | x) p(x)}{p(y)}.$$  

(2.34)

2.A.4 Hidden Markov Process

Bayesian estimation attempts to estimate the underlying signal of a Markovian hidden state process $x$, using the available observations $y$. In this context, the label "hidden" means that the process is unobservable, and therefore has to be estimated from indirect conditionally independent observation. Both $x$ and $y$ can be regarded as stochastic variables with new outcomes at each new $t$, resulting in the samples $x_t$ and $y_t$. Thus, the outcome of stochastic processes $x$ and $y$ can be described by their discrete sampled sets \{ $x_t ; t \in \mathbb{N}$ \} and \{ $y_t ; t \in \mathbb{N}$ \} respectively.

The state process $x$ is said to be Markovian, meaning that given a present state $x_t$, the future state $x_{t+1}$ is conditionally independent of the past. In other words, the probability of the next state $x_{t+1}$ assuming one certain value only depends on the value of the present state $x_t$. This gives the equation (see e.g. [BETA 88])
\[ p(x_{t+1} = a_{n+1} \mid x_t = a_n, x_{t-1} = a_{n-1}, \ldots, x_0 = a_0) = \\
= p(x_{t+1} = a_{n+1} \mid x_t = a_n), \quad (2.35) \]

or, differently written,

\[ p(x_{t+1} \mid x_t, x_{t-1}, \ldots, x_0) = p(x_{t+1} \mid x_t). \quad (2.36) \]

[Dou 98] gives a quite compact description of the unobservable (hidden) Markov process \( x \) and its corresponding conditionally independent observation \( y \). Given that \( p(x_0) \) is the initial distribution of \( x \), the entire process is described by the hidden Markov model

\[ p(x_0) \quad \text{and} \quad p(x_{t+1} \mid x_t) \quad \text{for} \quad t \geq 0 \quad (2.37) \]

\[ p(y_t \mid x_t) \quad \text{for} \quad t \geq 0. \quad (2.38) \]

### 2.B The Update Expressions

The measurement update \( p(x_t \mid \mathbb{Y}_t) \) and the time update \( p(x_{t+1} \mid \mathbb{Y}_t) \), given by (2.7) and (2.9) respectively, are derived using a combination of Bayes' theorem, the conditional independence between \( \mathbb{Y}_t \) and \( x_{t+1} \) and a manipulated form of the standard expression for conditional probability. The latter is derived starting with

\[ p(a, b) = p(a \mid b)p(b). \quad (2.39) \]

If another variable \( c \) is introduced and regarded as conditionally given, the corresponding expression becomes

\[ p(a, b \mid c) = p(a \mid b, c)p(b \mid c). \quad (2.40) \]

#### 2.B.1 The Time Update

The first step to derive (2.9) is a straightforward rewriting using (2.40). The second step uses the fact that \( x_{t+1} \) is conditionally independent of \( \mathbb{Y}_t \), when \( x_t \) is given. This is the case, as \( x_t \) is given by (and therefore also encloses all the information from) the entire chain of observations \( \mathbb{Y}_t \) up until time \( t \). The resulting expression

\[ p(x_{t+1}, x_t \mid \mathbb{Y}_t) = p(x_{t+1} \mid x_t, \mathbb{Y}_t)p(x_t \mid \mathbb{Y}_t) = p(x_{t+1} \mid x_t)p(x_t \mid \mathbb{Y}_t) \quad (2.41) \]

is integrated on both sides with respect to \( x_t \). This finally gives that

\[ p(x_{t+1} \mid \mathbb{Y}_t) = \int_{\mathbb{X}_t} p(x_{t+1} \mid x_t)p(x_t \mid \mathbb{Y}_t)\,dx_t. \quad (2.42) \]
2.B.2 The Measurement Update

The deriving of (2.7) starts with Bayes’ theorem. In the second step, the notational observation $\mathcal{Y}_t = \{ \mathcal{Y}_{t-1}, y_t \}$ is used. The third step applies the rewritten form (2.40) of the conditional probability. After that, the independence of $x_{t+1}$ with respect to $\mathcal{Y}_t$ if $x_t$ is given, followed by Bayes’ theorem, results in (2.44).

$$
p(x_t | \mathcal{Y}_t) = \frac{p(\mathcal{Y}_t | x_t) p(x_t)}{p(\mathcal{Y}_t)} = \frac{p(y_t, \mathcal{Y}_{t-1} | x_t) p(x_t)}{p(y_t, \mathcal{Y}_{t-1})} = \frac{p(y_t | x_t, \mathcal{Y}_{t-1}) p(\mathcal{Y}_{t-1} | x_t) p(x_t)}{p(y_t | \mathcal{Y}_{t-1}) p(\mathcal{Y}_{t-1})} = \left( \text{Bayes' Theorem} \right) = \frac{p(y_t | x_t) p(x_t | \mathcal{Y}_{t-1})}{p(y_t | \mathcal{Y}_{t-1})}.
$$

Hence, the resulting expression is given by

$$
p(x_t | \mathcal{Y}_t) = \frac{p(y_t | x_t) p(x_t | \mathcal{Y}_{t-1})}{p(y_t | \mathcal{Y}_{t-1})}.
$$

(2.44)
3 Underwater Terrain Navigation

A terrain aided navigation system would naturally require a lot more than merely a terrain navigation module. As the name implies, terrain aided navigation is not a self-sufficient system at this stage, but instead designed to support the traditional navigation system. This chapter describes a concept that takes advantage of any existing AUV sensors and thus inexpensively creates a compact and uncomplicated terrain-positioning module.

To begin with, this chapter introduces the conceptual Terrain Navigation System (TNS), around which the main part of this study evolves. It also accounts for some practical aspects of underwater acoustics, concerning the fixed single narrow-beam sonar model. Finally, the theory from Chapter 2 is applied to the terrain navigation problem, defining a comprehensive terrain-positioning particle filter framework.

3.1 Traditional Navigation Systems

Most traditional navigation systems are based mainly on dead reckoning. Starting at a known location, a position estimate is continuously calculated, possibly also accompanied by fixed-point calibration at other known locations. For this, a combination of sensors measuring kinetic state, velocity and travelling depth is used. The backbone of the traditional Inertial Navigation System (INS) is the Inertial Measurement Unit (IMU), measuring the kinetic state of the vehicle. Based on gyros and accelerometers, it detects acceleration changes in three dimensions (Figure 3.1).

Additional sensors, e.g. sensors estimating the velocity or pressure-sensors used to estimate the travelling depth, are also needed. The vehicle velocity, compared to the surrounding water, is often estimated as a function of the rpm of the propulsion system. A more accurate way to determine either the absolute velocity compared to the seabed or the relative velocity against the surrounding water is to use a Doppler log. This is an active sensor measuring the Doppler phase-shift in a sound pulse reflected against the

![Figure 3.1](image)

*Figure 3.1 The AUV co-ordinate system. Turning around the axes x, y and z is called roll (x), pitch (y) and yaw (z) while the translation is called surge, sway and heave respectively.*
bottom or the surrounding water. However, due to its active nature, the Doppler log might be untimely to use in many situations.

3.2 The Conceptual Terrain Navigation System

When studying underwater terrain navigation, the possible requirements of a future AUV are highly interesting. One example of this kind of vehicle is the Torpedo Mine Sensor (TMS) concept, currently being developed at SAAB Bofors Underwater Systems. This multiple-role vehicle is designed for both survey and reconnaissance missions as well as for combined torpedo/mine missions. It should be able to operate with or without support from any outside systems, such as submarines or surface-vessels. As briefly described in Chapter 1, the demand for high positioning accuracy during such extended missions requires some kind of support for the INS of the AUV. This aid could be provided by a TNS.

Even though a TNS often would provide a vital increase of the positioning accuracy, it is not a self-sufficient system. It will require access to information such as velocity, course and travelling depth, provided by sensors or navigation sub-systems that are normally part of a traditional INS. As implied in the introduction to this chapter, the purpose here is not to describe how to integrate a terrain navigation module in an autopilot, but to generally investigate the possibility of using the terrain for positioning. Therefore, no effort has been made to analyse the impact of a TNS position estimate on the vehicle control. Neither has any aspects or difficulties surrounding the integration of such systems been examined. Instead, the modelled vehicle is assumed to use INS data for its navigation, providing more or less accurate sensor-readings to the supporting TNS system. A principal description of how sensor information could be acquired and shared between subsystems is given by Figure 3.2. This general framework will from now on be referred to as the conceptual Terrain Navigation System model or, in short, the conceptual TNS model.

![Figure 3.2](image)

*Figure 3.2 The TNS receives information about momentary course, velocity, travelling depth and measured depth from surrounding systems.*
3.3 Navigation Requirements on Underwater Terrain

There are quite a few requirements that have to be met concerning the seabed, in order for it to serve as navigation material. Some of the main requirements are listed below.

- The terrain must not be significantly altered over time. This could rule out some bottom areas consisting of sand or sediment if they are affected by shifting sea currents or if new sediment is constantly added, e.g. at a mouth of a river.

- The terrain elevation must contain some variation. This would rule out large non-coastal areas of the Baltic Sea, where the sea bottom is more or less a plane field.

- Naturally, there must already exist an accurate and detailed map covering the area. As for now, these types of high-resolution seabed charts are relatively rare, even though their numbers are increasing as results of ongoing charting projects.

In order to circumvent these obstacles, the terrain navigation could be used exclusively within navigation cells. These are areas where the bottom’s structure is invariant over time, contains varying depths and is accurately charted. Between these navigation cells the INS will lack support from the terrain positioning system, but once inside the navigation cells the terrain information could provide a vital position-update.

One important step towards implementing an actual TNS is to define a measure describing the achievable TNS navigation accuracy, provided some specified conditions such as resolution and accuracy of the map, quality of sonar-measurements and terrain variation relative to map resolution etc. Due to a lack of time, this area has not been fully investigated here. Instead, the reader is referred to e.g. “Particle Filtering and Cramér-Rao Lower Bound for Underwater Navigation“ [Kar 02b], which describes the Cramér-Rao lower bound as such a measure. For that paper, R. Karlsson and F. Gustafsson were provided with the underwater map data described in Section 4.4.

3.4 Practical Aspects

As mentioned earlier, surfacing to calibrate the navigation system by receiving a GPS position-update is for most military applications not an option. Instead, many AUVs will have to manage with whatever navigational information is available undersea. For shorter missions, the accuracy provided by a dead-reckoning system is satisfying. However, problems start to occur when planning for longer missions, spanning over days or weeks. Over such a time span, a course drift-rate equivalent to 1°/h is large enough to completely loose orientation before the mission is over³. Here, the possibility of supporting the INS with terrain navigation seems appealing.

³ 1°/h (3σ) is a public value of the gyro drift-rate of the relatively accurate IMU used in e.g. Torpedo 62, given by [And 00]. In this context, this gyro drift-rate should be seen as a lower bound on what accuracy is possible to achieve.
3.4.1 Cost per Payload

For all AUVs, ratios like cost per payload or weight per payload etc, are crucially important to meet the requirements set for the product. The desire is always to maximise the possible payload for each application. This also affects the choice of sonar.

The three-dimensional image given by a 3D-sonar (using multiple or moving wide or narrow sonar beams) could naturally provide much information about the terrain at any given moment. Still, the simplicity of the fixed single narrow-beam sonar makes it highly interesting. An educated guess would tell that a TNS based on such simple sonar would be cheaper and easier to produce than one with a 3D-sonar. Due to the nature of the fixed beam, its placement inside the vehicle could most likely be chosen more freely and it would normally be smaller and lighter.

3.4.2 Sonar Beam Refraction

Many depth-measurement errors originate from refraction phenomena due to layers of water with different temperature and/or salinity. These layers can cause both reflections, giving the location of the layer as a false depth reading, as well as changing the travelling path of the sonar-beam, causing the depth to be measured at a different location than expected. When not aiming the sonar-beam perpendicular at the bottom, this risk of false depth measurements is significantly increased, as illustrated by Figure 3.3 [Uri 83].

When using a beam aimed straight downwards, these effects are mainly avoided. The remaining possibility of false echoes caused by layer transitions can be handled much easier than refraction, e.g. by detecting the significantly weaker signature of a layer-reflected echo.

One could also note that a fixed sonar-beam, trained perpendicular against the bottom, would provide a smaller acoustic signature than a diagonal beam. It would therefore probably have a lower risk of being detected, since it would not spread out excessive sideways reflections from the bottom, as illustrated by Figure 3.4. Instead, most of the
energy would be reflected back towards the sender. Therefore, a single narrow-beam sonar aimed straight downwards would also, most likely, consume less energy than a sideways aimed sonar.

![Figure 3.4](image)

*Figure 3.4* A vehicle equipped with a fixed single narrow-beam sonar faces a lower danger of detection than one with a sonar aiming sideways. This is due to the sideways-propagated reflections, increasing the vehicle acoustic signature.

### 3.5 Single Beam Terrain Navigation

As described in the previous sections of this chapter, a terrain navigation based on a fixed single narrow-beam sonar seems to have many advantages, despite the fact that it cannot provide any three-dimensional image of the surrounding area. The idea is instead to continuously measure the distance to the seabed and thus create a terrain elevation track or a terrain profile of the terrain beneath the vehicle route. A momentary position estimate is recursively obtained by comparing the measured terrain profile to the terrain database. An obvious downside with this concept, compared to a concept with a 3D-sonar, is that the amount of information only increases as long as the vehicle moves ahead. However, this sonar model, illustrated by Figure 3.5, will be used in the following chapters.

Unfortunately, terrain navigation based on a fixed single narrow-beam sonar requires a relatively high on-line computational capacity in order to make a reliable real-time position estimate. Fortunately, the rapid development of embedded system performance makes this a problem of decreasing significance. In addition, any map matching of a 3D-sonar image would also result in a heavy computational burden. (See for instance the approach made in [Nyg 99].) Thus, the computational requirements might be similar to many other approaches.

#### 3.5.1 Depth Notations

The depth stored in the terrain database would most likely be the distance from surface to seabed, further on referred to as the terrain depth \( h \). As seen in Figure 3.5, the difference between the measured depth \( y \) and the terrain depth \( h \) is given by the
travelling depth $d$. In order to calculate the difference between measured depth and terrain depth, the AUV must be able to determine its current travelling depth. This could be done with satisfying accuracy using a pressure sensor or a single sonar-beam trained at the surface\(^4\). Even after this adjustment, there will still remain a difference $\varepsilon$ between measured depth, travelling depth and terrain depth, originating from different sources. Some will be introduced by limited map resolution, forcing interpolation between grid points. Others will occur from inaccurate terrain depth values stored in the terrain database or from errors in the measured depth. This difference is defined as

$$\varepsilon = y - h + d.$$  \hspace{1cm} (3.1)

\[\text{Figure 3.5} \quad \text{Depth measurement by a fixed single narrow-beam sonar. The measured depth } y \text{ should be the difference between the expected depth according to the terrain database } h(x) \text{ and the travelling depth } d.\]

---

\(^4\) This could be done by many subsurface applications, using an existing proximity fuse measuring the vertical distance to the surface.
3.6 Recursive Bayesian Terrain Navigation

The recursive state model from Chapter 2 naturally requires some application-specific modification before it could actually be used. Such changes are made in this section, defining the TNS state model. This section also contains a short description of a particle filter, based on these changes.

3.6.1 The TNS State Model

In Section 2.1, a general state model for a hidden state process $x$ was given by the state transition equation (2.3a) the corresponding measurement equation (2.3b), written as

$$
\begin{align*}
  x_{t+1} &= f_t(x_t, v_t) \\
  y_t &= h(x_t) + e_t \\
  t &= 0, 1, 2, \ldots
\end{align*}
$$

(3.2)

A recapitulation gives that $y_t$ is the sampled observation received at time index $t$ and that $x_{t+1}$ is the predicted next state, based on the current state $x_t$ and the most recently received observation $y_t$. In the simplest case of terrain-aided underwater navigation, the state variable $x$ is two-dimensional and describes the horizontal position of the vehicle. As indicated by [Ber 99], it is possible to increase the state dimension. The state could, for instance, also include information about travelling depth bias or INS drift. According to [Ber 99], this could be done without critically increasing the computational burden. Here, however, the assumption is made that the surrounding subsystems are able to provide sensor data, such as travelling depth, with sufficient accuracy.

When comparing (3.2) to the variables and parameters previously given in Section 3.5 and Figure 3.5, the observation $y_t$ and the function $h(x_t)$ naturally corresponds to the measured depth beneath the vehicle and the terrain depth at that position respectively. In Section 2.3, the residual $e_t$ was described as a stochastic variable with a more or less known probability density $p_{e_t}(\cdot)$. Here, the properties of $p_{e_t}(\cdot)$ mainly depends on the accuracy of the following four quantities: the measured depth measurement, the travelling depth measurement, the accuracy of the terrain database values and, finally, the resolution of the terrain database. A large resolution could possibly result in larger calculation errors. These are always more or less present, since the depth at any position $x$ must be approximately calculated from the terrain database, using interpolations between the depth values specified in the surrounding grid-points. This could potentially result in large approximation errors if the map resolution is too sparse to capture all the significant terrain variation.

When comparing the residual $e_t$ to the description given in Section 3.5, $e_t$ could be interpreted as a combination of the travelling depth $d$ and the difference $\varepsilon$, given by (3.1), as

$$
\varepsilon = y - h + d.
$$

(3.3)
Hence, $e_t$ becomes

$$e_t = e_t - d_t = y_t - h(x_t), \quad (3.4)$$

provided that $x_t$ is the actual position where the depth measurement $y_t$ is made. Consequently, the stochastic property of the residual $e_t$ would have an expectation value $E(e_t) = -d_t$, where $d_t$ is the travelling depth at time index $t$. From now on, the travelling depth will no longer be explicitly stated. Instead it is regarded as a parametric input to the algorithm, taken care of by constantly adjusting the output from the terrain database to compensate for the travelling depth impact on (3.4).

So far the transition equation has been kept general, but when applying the state model (3.2) to the conceptual AUV model, a more specific description is given by

$$\begin{cases} x_{t+1} = x_t + u_t + v_t \\ y_t = h(x_t) + e_t \end{cases} \quad t = 0, 1, 2, \ldots \quad (3.5)$$

Here, the general transition function $f_t$ is specified as

$$f_t(x_t, v_t) = x_t + u_t + v_t, \quad (3.6)$$

where $u_t$ is the movement vector given by the velocity and course provided by the surrounding navigation sub-systems. The movement vector $u_t$ is regarded rather as a

![Figure 3.6](image)

*Figure 3.6* The movement vector $u_t$ and its projection into the plane, which describes the pdf of the predicted state $x_{t+1}$. The location of $x_{t+1}$ depends on $p_d(\cdot)$, which in its turn depends on the nature of the uncertainties in course and velocity.
parametric input signal than as an actual measurement. This is motivated by the more accurate measurements of momentary course and velocity compared to the position knowledge. Therefore, $u_t$ is regarded as a more or less known input parameter, affected by a stochastic process noise $v_t$, and is consequently not a part of the state vector.

Depending on the accuracy and orientation of $u_t$, the properties $p_{v_t}(\cdot)$ of the noise component $v_t$ are determined. As illustrated by Figure 3.6, the course-uncertainty will give a spreading of $v_t$ perpendicular to the movement vector $u_t$, while the velocity-uncertainty will give a spreading along $u_t$. Naturally, the effect of the course spreading is proportional to the size of $u_t$. Whether or not the effect of the surgewise uncertainty is proportional to the size of $u_t$ depends on the nature of its origin, such as if it is a constant measurement noise or a velocity related system noise. Anyhow, the properties of the noise $v_t$ depends on the momentary size and orientation of $u_t$, even if this dependence is more or less suppressed in equation (3.5) above. One could argue, though, that at this level of abstraction, such simplifications are entirely appropriate. At this stage, it is better to avoid unnecessary connections between the more or less general TNS state model and some future unspecified navigation sub-systems. However, the actual implementation of the particle filter used in Chapter 5 will take this into account.

### 3.6.2 The TNS Particle Filter

At this stage, all necessary components of a terrain navigation particle filter have been described. Both the transition equation (used for propagating the particles) and the measurement equation (used for evaluating their importance-weights) are given by (3.5). Here, the state vector $x$ is two-dimensional, describing the position in northwards and eastwards co-ordinates. Likewise, the movement vector $u$ gives the momentary horizontal vehicle movement. As mentioned before, $u$ is regarded as parametric input affected by the process noise $v$. The main characteristics of $v$ naturally depend on the sensors used for measuring $u$, and could naturally also depend on the momentary orientation of $u$.

The measured depth is given by $y$, and the terrain database, adjusted for the current travelling depth, gives the expected depth at the location of $x$. The difference in-between, the residual $e$, is, hopefully, relatively well known for the system at hand. If the outcome of $e$ is stored in an error histogram, the knowledge about characteristics of $e$ would, naturally, increase over time. Another approach would be to regard the properties of $e$ as design parameters. If the true characteristics of $e$ are not entirely known, a number of simulations with different residual distributions could be made. This could, in a black-box kind of way, give a good hint of the optimal estimate of $p_e(\cdot)$ under given conditions.

Now, the SIR particle filter implementation given in Section 2.5.4 is easily adapted to fit the terrain navigation problem. More details about how this can be done for simulation use is found in Chapter 5.
4 Depth Charting and Map Generation

When evaluating any algorithm, the access to actual experimental data often makes all the difference. With this in mind, a small area of the Swedish lake, Lake Vättern was charted to provide an actual underwater terrain map. The created depth map is mainly intended for navigation algorithm evaluation. In this way, the chosen navigation algorithm was tested not only on simulated map-material, but also on actual terrain data.

The charted area was chosen in the immediate vicinity of SAAB Bofors Underwater Systems’ company location in Motala. Along with the cartography of the area, a couple of evaluation tracks crossing the charted area were made separately. The evaluation track data is hence not a part of the data material used to generate the map. In order for the reader to assess the significance of the navigation results from the generated map, this chapter will contain a general description of the charting process, the equipment used and the nature of the approximations and data pre-processing made before generating the final map.

4.1 Aim and Practical Approach

The aim of the charting process has been to cover an area of approximately 300 by 300 meters with parallel depth-measurement tracks separated by five-meter distances. Since the equipment available for the measurements only measures the depth within a relatively small area directly below the device, this charting approach was believed to give coverage sufficient enough to create a depth chart with a five by five meters horizontal resolution. Although the original intention was to gather the depth data by travelling back and forth along parallel five-meter corridors, the practical aspects of steering a boat within such parallel corridors turned out to be more of a challenge. The

Figure 4.1 Schematic description of the measurement equipment. The laptop logs and presents the acquired data from the GPS receiver and the altimeter.
resulting measurement tracks tend to lie more closely or even cross each other in some regions, while other parts of the map areas were more sparsely covered.

4.2 The Measurement Equipment and Software

This section will give a thorough description of the measurement equipment and the software used for data logging and presentation, schematically described by Figure 4.1. One aim of this description is to make it possible for the reader to assess the significance and accuracy of the derived map, and thus also the navigation results from simulations made on data derived from those depth soundings. Another aim is to describe what kind of results that are possible to achieve with relatively simple means, and perhaps also to give some hints to others facing similar projects.

4.2.1 The Measurement Platform

One of the most important choices preceding the measurement runs was the choice of the measurement platform, i.e. the boat from which the measurements were made. This would also be the single most expensive factor during the entire charting procedure, which naturally is not an issue to be neglected when running a project within a commercial company. It would be natural to strive after a measurement platform as large as possible in order to reduce the effects of sea rolling, which is always more or less present even on a calm day. Excessive sea rolling or tilting from an unsymmetrical load or moving crewmembers could severely cripple the resulting depth readings, unless the tilt-angle was continuously measured. Rather early in the preparation stages of the depth soundings, the idea with tilt-angle measurement was abandoned due to the complex technical difficulties surrounding such an approach. Instead, it was decided to make the depth soundings on a relatively calm day, and consequently chose a boat large enough to withstand the rolling but still not too expensive to use.

The choice of measurement platform finally fell on the twin-engine, 6.2 tonnes displacement, 28-foot boat UW4 (Figure 4.2) owned by SAAB Bofors Underwater Surveillance AB

![Image of UW4 boat](image-url)  
*Figure 4.2 The surveillance boat UV4 was used during the soundings. Though it is mainly constructed for high speed, it provided a stable platform travelling 2 to 3 knots.*
The main use of this boat is surveillance during torpedo test firings, but its large open rear deck, in combination with its high manoeuvrability and its relatively good rolling stability, made it the most natural choice for this assignment.

As many boats of this size, the UW4 is not built to have an optimal rolling stability in speeds between two and three knots. This speed interval, however, was the assessed optimum speed interval, considering both the desired number of position measurements within a five-by-five meter square as well as the stability of the altimeter mounting rack (See the description further on in this section). In the end, most of these educated guesses turned out to be accurate enough, and the UW4 made a sufficient and reliable measurement platform. Even when the wind started to make the surface a bit rougher than what was considered to be ideal conditions, this was still the case.

4.2.2 The Data Logging Hardware

One of the most central factors for the quality or the received measurement data was the performance of the laptop computer, an IBM ThinkPad with Windows 2000. This PC was used for data logging and travelled route presentation. Even though the data rate and volume perhaps was not that high, the PC was still required to receive and store data from three serial ports simultaneously. The relatively poor real-time performance of Windows 2000, causing message buffering and delays, initially caused some trouble but after changing priorities on the processes used for monitoring the ports, most of these obstructions were circumvented.

4.2.3 The Altimeter and the A/D converter

The sonar used for the depth-measurements was the standard single-beam altimeter Simrad Mesotech 807 Echo Sounder. This altimeter is, for instance, used in some of the company’s ROV products (Remotely Operated Vehicle) to measure hovering altitude over the seabed. Some of its specific data is given in Table 4.1. The relatively high frequency of the 807 Echo Sounder enables it to make accurate measurements more or

![The Simrad Mesotech 807 Echo Sounder (Left) was mounted on a steady rack (Right), placing the measurement plane about one meter below the water surface.](image)
less regardless of the nature of the seabed. This could be a problem with some sonars using relatively low frequencies. These easily penetrate the softer upper layers of the seabed and are instead reflected some distance further down, when striking the more solid parts of the seabed.

Another advantage of the 807 Echo Sounder is its narrow transducer (transmitting and receiving) cone. The narrower the listening cone is, the more accurate the received measurements will be. Depending on the nature of the echo detection (threshold, peak-detection etc.), an average depth estimate over the area reflecting the beam will be made in the sensor. With a transducer cone of 1.7°, the area reflecting the beam at the maximum measured depth during the charting process (roughly 20 meters) corresponds to a circle with less than 0.3 meter radius. Taking this into account, the ideal 8 mm resolution of the altimeter is more than accurate enough for these measurements.

<table>
<thead>
<tr>
<th>Simrad Mesotech 807 Echo Sounder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
</tr>
<tr>
<td>Range</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
</tbody>
</table>

Table 4.1 Parameter data of the altimeter used in the depth soundings.

<table>
<thead>
<tr>
<th>Advantech ADAM-4012 Analog Input Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling rate</td>
</tr>
<tr>
<td>Input range</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
</tbody>
</table>

Table 4.2 Parameters of the A/D-converter used to read altimeter output voltage.

The output signal of the 807 Echo Sounder is an analogous voltage between 0.20 and 10.00 volts, corresponding to a depth of 2 and 100 feet (0.61 and 30.50 meters) respectively. Before the analogous output voltage could be stored in a data log file it was sampled by an Advantech ADAM-4012 A/D converter, specified in Table 4.2. The sampling rate during the measurements was set to 10 Hz, which can be regarded as an adequate compromise between the desire for a large data set and the Windows 2000 real-time data logging capacity. The accuracy of the ADAM-4012 A/D converter is ±0.05% or better, which corresponds to an 1.5 mm depth resolution at 30.5 meters depth, i.e. significantly below the resolution of the altimeter itself.

4.2.4 The GPS Receiver

During the deep-sea sounding, a GPS receiver was used to keep track of the sounding position. Since the general function of the Global Positioning System nowadays can be regarded as common knowledge, no detailed description of the GPS system will be given here. Instead, the interested reader is referred to any of the numerous comprehensive reports and books written on this topic, e.g. [Lin 02].

40
The GPS used during the charting procedure was the Ashtech SCA-12S GPS receiver (Figure 4.4). Even though this particular model now is a few years old, the stand-alone accuracy of the SCA-12S is 16 meters rms or less\(^5\) if the satellite coverage is sufficient enough. A number of four satellites or more would be enough to achieve that accuracy, which was the case throughout the entire series of measurements.

One of the advantages of the SCA-12S is its ability to send and receive information on three serial ports simultaneously. These serial ports are denoted Serial A, B and C. This feature was used to provide the data logging program with position messages on Serial C, while the real-time navigation and tracking application at the same time received the corresponding information for on-screen presentation on Serial A. Parallel to this, Serial B was used to receive position corrections from the DGPS receiver.

The log file output from the SCA-12S was tapped through Serial C as strings of ASCII characters, using the POS (position) message specified by the message format NMEA 0183 Standard Version 2.0. The on-screen application GPSU (see below) received its positioning information as a NMEA GLL-message on Serial A. The difference between these two message formats and a description of their different data fields is given by appendix 4.A. The output rate for both serial ports was set to 1Hz, using the NMEA PER-message.

In order to achieve a positioning accuracy sufficient for a five-by-five meters map resolution, i.e. preferably accuracy below one meter, the stand-alone performance of the SCA-12S (or, for that matter, most other GPS devices) was not good enough. Hence, the GPS position was supported by a DGPS correction signal, which is constantly transmitted as an RDS signal on the Swedish national public service radio channel P4. The SCA-12S was set to use this Differential GPS correction when available, which turned out to be most of the time, but still provide a GPS position, even if the correction signal was lost. Whether the position is differentially corrected or not is flagged in the

---

\(^5\) The 16 meters rms error value is given by the Ashtech SCA-12S operating manual. However, when using differential correction, the accuracy of the SCA-12S is better than one meter rms.
POS message, making it possible to assess the significance of each given position. The accuracy of the SCA-12S is better than one meter rms when differentially corrected.

4.2.5 The DGPS Receiver

As mentioned in the previous section, the positioning accuracy of the SCA-12S is approximately 16 meters rms without supporting systems. One example of such a supporting system is the Differential GPS (DGPS) system, which is based upon a number of fixed GPS receivers located at well known locations all over the country. Each of these GPS receivers continuously compare their calculated GPS position to their known position and passes the difference on to a central computer. Based on the correction information from these receivers, a differential correction is calculated and transmitted as a RDS-signal over the public service broadcast radio network.

Since the SCA-12S GPS receiver does not contain a DGPS receiver itself, the external Aztec Radiomedia Differential GPS receiver RXMAR1 was used (Figure 4.4). During the depth sounding it constantly provided the GPS receiver with corrections, increasing the positioning accuracy of the SCA-12S well below one meter. For more comprehensive information on the DGPS technique EPOS and the Swedish provider, see e.g. the homepage of Cartesia Informationsteknik AB.

4.2.6 The Equipment Rack

The altimeter was mounted on a steel rack located on the starboard side of the boat, placing the altimeter measurement plane around one meter below the surface aimed straight at the bottom. The location of the rack is shown in Figure 4.5. On top of the rack, the antennas for the GPS signals and the DGPS correction RDS signal are mounted.

4.2.7 The Data Logging Software

During the charting process, the position and depth data was sampled at different sampling rates and sent to separate serial ports on the laptop. The program used to request and store received position- and depth messages is the easy to use Multi-threaded TTY (MTTTY) program, which
elegantly handles communication through the serial ports of the computer (Figure 4.6).
The program monitoring the port receiving GPS position messages was set to receiving only, thus storing the incoming POS messages in a position log file as described in appendix 4.B. The program monitoring the depth information serial port had to be set to request a new reading from the A/D converter every 0.1 seconds while storing the readings in a log file.

In order to combine the corresponding position- and depth readings, a time tag containing the internal system time was attached to each sample. In order to do this, the MTTY had to be modified to attach the system time at the beginning of each position and depth message, before storing it in the log files. This gives a highly accurate specification of the relative arrival-time of the different messages, since the system time is given with an accuracy of approximately ±0.5 ms.

When determining which routes had already been covered in previous measurement runs, the desktop application GPS Utility (GPSU) was used for real-time on-screen travelled route presentation (Figure 4.7). Thus, this program served as the primary navigation aid during the charting procedure. The GPS information used by GPSU was received as GLL messages on the third serial port (Serial B) with an update rate of one position per second. Thanks to the applications ability to store its own track logs, the position data used for navigation could be stored separately from the raw POS message data log. During the entire charting process, the GPSU feature of optional hiding and high-lightening of previously travelled routes or still uncharted areas made it relatively easy to control which routes should be next, or where to continue the charting process on a later occasion.
Due to technical shortcomings, as well as actual disturbances, the acquired depth data needed some editing and/or filtering before it could be used to generate a map. Fortunately, the corresponding position data was generally not affected by such errors. The two main categories of erroneous depth data samples, outliers and logging errors, are both described in detail in this section.

### 4.3 Resulting Data

Figure 4.7: A small screen-shot from the program GPS Utility, used for singling out areas still uncharted.
4.3.1 Depth Data Outliers

Here, a depth measurement is considered to be an outlier when the depth change between two samples is obviously too large to have a logical physical explanation, or if the depth value falls outside the altimeter range. Some examples of outliers are given in Figure 4.8, where the zero meters and the sudden 30 meters readings obviously are not seabed readings. The origin of one specific outlier is difficult to determine, but some general observations have been made.

Many of the outliers occurred when the boat was turned around. During these turning manoeuvres, the propulsion rate was frequently increased to turn the boat around faster. Sometimes the port and starboard engines were even thrust in opposite directions to increase the turning rate. All this industrious propeller use naturally caused massive turbulence in the water, which may have resulted in false echoes due to refraction from the turbulence, or from air bubbles drawn down into the water surrounding the altimeter. However, most of this turning and manoeuvring was made outside the intended map area, and would, hence, not have affected the resulting depth chart anyway. Still, the resulting obviously false depth readings have been removed from the data set.

The outlier removal was mainly made using simple threshold values and pattern recognition. If a depth reading i.e. was not between specified minimum and maximum values or if the difference relative to the surrounding depth readings exceeded a specified delta value, the depth reading was simply considered false and discarded.

4.3.2 Depth Data Logging Errors

The logging errors occurred when the laptop operative system could not manage to provide a real-time data logging capacity. As mentioned earlier, a slightly modified version of the MTTTY program was used to tag each depth and position reading with the laptop system time, before storing it in a log file. Thus, the real-time performance of the operating system was crucial for the quality of the result.

Unfortunately, Windows 2000 is not a very reliable real-time operating system, and even after increasing the priority of the MTTTY program thread, some depth readings were buffered before being passed on to the MTTTY program. The result of this buffer delay was that a consecutive number of depth readings (normally around 20) were stored without being time-tagged. A typical example of this is given in Figure 4.8.

The relative order of the samples was still valid, since the buffer (to the best of our knowledge) would not alter the sequence of received messages, but merely add some delay and then release the messages simultaneously. This simultaneous release of delayed messages would have been hard to detect, if it had not been for an integrity check of the relative arrival-time of the messages to the MTTTY. The modification of the MTTTY was designed to only time tag messages arriving with a valid time difference compared to the previous message. In other words, if the sampling rate of the
depth data was set to be 10Hz, messages arriving with a time gap significantly lower than 0.1 seconds were considered unreliable and hence not time-tagged at all. These messages were still stored, though, and their approximate sampling time could be assessed later, as illustrated by the example above.

Most of the false or missing time tags could be estimated relatively accurately, as the nearest surrounding correctly time-tagged depth readings are easily identified. The time gap between these samples was compared to the expected corresponding time-span given by of the number of depth readings lacking time tags. As long as this comparison did not differ too much, a linear interpolation between the true surrounding time tags was made, providing the samples missing time tags with new ones.

### 4.3.3 Position Data Quality

For some reason, the position data did not suffer from any logging errors of such nature as the depth data was affected by. This could perhaps be referred to either as luck or as a consequence of a lower data rate (1 Hz compared to 10 Hz). Nevertheless, the general quality of the received position data appears to be rather high. Almost no position outliers or jumps relative to the consecutive line of position measurement can be detected. Also, the number of satellites used by the GPS receiver was for most of the time closer to ten than to four, which is the minimum acceptable number of satellites providing an acceptable positioning accuracy. Neither did potential problem from difficulties of receiving the DGPS correction signal occur at all. Over all, the general appearance of the received position data coincides well with the expected below-one-meter accuracy, which should be the case for the SCA-12S using differential correction.

![Figure 4.8](image)

*Figure 4.8  When the system time is plotted against arrival order, the samples typically occur like above. Two blocks of approximately 30 samples each have been delayed. Most of the delayed samples have missing time tags, but the first in each series receive a false time tag. New time tags are interpolated for both false and missing ones.*
4.4 Map Generation

Since the position data and the depth data were sampled at different sampling rates (1 Hz and 10 Hz respectively), an interpolation estimating the location of the depth readings between two consecutive position samples was necessary. The relatively low speed (2-3 knots) of the measurement platform resulted in an average maximum travelling distance of 1.5 meters between two position readings. It is reasonable to assume that the boat held an approximately straight course between any two consecutive position readings. Therefore, a simple linear interpolation, with respect to the system time, was made to assign an approximate position to each depth reading. When the interpolated track points are plotted in three dimensions, Figure 4.9 is received.

As seen in Figure 4.9, many of the measurement tracks are quite close to each other. The tracks also cross each other frequently. Other areas within the map area are not that well covered by measurement tracks. This makes the choice of the map resolution a bit more difficult. The original intention was to create a map with five by five meters resolution, but as seen in the track plots, some areas are not sufficiently covered to justify this resolution. Still, other areas are far better covered.

*Figure 4.9* The complete set of depth soundings after depth outlier removal gives an illustration of the degree of area-coverage. The rectangles mark the chosen map area. (Note that the two long straight lines are not measurement tracks, but simply lines connecting different log-files.)
In order to be able to use the additional information about the well-covered areas, a map resolution of one by one meter was chosen. This way, the more precise knowledge of some areas will be put to use. As long as no severe interpolation errors are made when converting the track-based data set into a grid-based depth chart, no additional inaccuracies will be introduced in the less known areas, than would have been the case if choosing a five by five meter's grid.

Some considerable effort was made to determine how best to transfer the depth tracks into a more map like grid-based structure. Finally, the choice conveniently fell on the MATLAB standard function `griddata`. This is a function especially designed to convert data specified at an irregular grid into data specified at a desired regular grid. Four different interpolation methods can be used by `griddata`, but to minimise the introduced interpolation errors (for reasons mentioned above), a linear interpolation method was chosen.

In Figure 4.10 a three-dimensional plot of the resulting depth map is presented together with the evaluation tracks made for navigation simulations. As mentioned earlier, these evaluation tracks are not part of the data set used for map generation. When viewing the map area from above, it appears to be relatively flat. However, the depth variation becomes more evident when the map is viewed using a more sea-chart-like approach, as in Figure 4.11. Here, the depth curves are separated by 0.2 meters depth difference. (Also see Figure 5.10.)

![Figure 4.10](image)

*Figure 4.10  The depth soundings within an area of 305 by 475 meter's were singled out to create the final map. The MATLAB command `griddata` was used to transform the large set of scattered data to a regular one by one meter grid, using linear interpolation.*
Figure 4.11 The Resulting depth-chart. The distance between the equidistant lines is 0.2 meters, as indicated by the depth numbers on the right side of the depth chart.
Appendix

4.A  NMEA-0183 Messages Formats

Here follows a short description of the two main output messages used during the depth chartings. Both messages are specified according to the NMEA-0183 standard\(^6\). The first is the POS message used for position logging, the other is the GLL message used by GPS Utility for on-screen presentation.

4.A.1  The POS Message

The POS message response format is specified as:

```
$PASHR,POS,n,qq,hhmss:ss,ddmm.mmmmm,s,s,dddmm.mmmmm,s,aaaaaa.aa,seeeee,ttt,
  ggg,svvv,pp,tt,tt,vvv*cc<CR><LF>
```

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>NMEA String Separator</td>
</tr>
<tr>
<td>PASHR</td>
<td>Header (Property of Ashtech)</td>
</tr>
<tr>
<td>POS</td>
<td>Message ID</td>
</tr>
<tr>
<td>n</td>
<td>0/1 corresponds to raw position / differentially corrected position</td>
</tr>
<tr>
<td>qq</td>
<td>Number of satellites used</td>
</tr>
<tr>
<td>hhmmss:ss</td>
<td>Current UTC time in hours, minutes and seconds</td>
</tr>
<tr>
<td>ddmm.mmmmm</td>
<td>Latitude in degrees, minutes and fractions of minutes</td>
</tr>
<tr>
<td>s</td>
<td>Latitude sector, E = East, W = West</td>
</tr>
<tr>
<td>ddddmm.mmmmm</td>
<td>Longitude in degrees, minutes and fractions of minutes</td>
</tr>
<tr>
<td>s</td>
<td>Longitude sector, N = North, S = South</td>
</tr>
<tr>
<td>aaaaaa.aa</td>
<td>s + or -, aaaaa.aa is computed altitude above WGS-84 reference ellipsoid</td>
</tr>
<tr>
<td>seeeee</td>
<td>Reserved</td>
</tr>
<tr>
<td>ttt</td>
<td>Course over ground (000 - 359 degrees)</td>
</tr>
<tr>
<td>ggg</td>
<td>Speed over ground (000 – 999 knots)</td>
</tr>
<tr>
<td>svvv</td>
<td>s + or - , vvv is vertical velocity (000 – 999 dm/s)</td>
</tr>
<tr>
<td>pp</td>
<td>PDOP, position dilution of precision</td>
</tr>
<tr>
<td>hh</td>
<td>HDOP, horizontal dilution of precision</td>
</tr>
<tr>
<td>vv</td>
<td>VDOP, velocity dilution of precision</td>
</tr>
<tr>
<td>tt</td>
<td>TDOP, time dilution of precision</td>
</tr>
<tr>
<td>vvv</td>
<td>SCA-12S Firmware version ID</td>
</tr>
<tr>
<td>*</td>
<td>Separator</td>
</tr>
<tr>
<td>cc</td>
<td>Checksum</td>
</tr>
<tr>
<td>&lt;CR&gt;</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>&lt;LF&gt;</td>
<td>Line Feed</td>
</tr>
</tbody>
</table>

Typical examples of received POS messages when taking the measurements is given by

---

\(^6\) NMEA - National Marine Electronics Association
### 4.A.2 The GLL Message

The GLL message response format is specified as:

```plaintext
$GPGLL,ddmm.mmmmm,s,dddmm.mmmmm,s,hhmmss.ss,s,A*cc<CR><LF>
```

<table>
<thead>
<tr>
<th>$</th>
<th>NMEA String Separator</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GPGLL</td>
<td>Header</td>
</tr>
<tr>
<td>ddmm.mmmmm</td>
<td>Latitude in degrees, minutes and fractions of minutes</td>
</tr>
<tr>
<td>s</td>
<td>Longitude sector, E = East, W = West</td>
</tr>
<tr>
<td>dddmm.mmmmm</td>
<td>Longitude in degrees, minutes and fractions of minutes</td>
</tr>
<tr>
<td>s</td>
<td>Longitude sector, S = South, N = North</td>
</tr>
<tr>
<td>hhmmss:ss</td>
<td>Current UTC time in hours, minutes and seconds</td>
</tr>
<tr>
<td>s</td>
<td>Status, A = Valid, V = Invalid</td>
</tr>
<tr>
<td>*</td>
<td>Separator</td>
</tr>
<tr>
<td>cc</td>
<td>Checksum</td>
</tr>
<tr>
<td>&lt;CR&gt;</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>&lt;LF&gt;</td>
<td>Line Feed</td>
</tr>
</tbody>
</table>

### 4.B Time Tags

As described in Section 4.2.7, a time tag is attached to each messages stored in the data log file. The time tag is given by the laptop internal system time, specified as a number where the least significant digit corresponds to 10e-4 seconds. The time tagging was made by a modified Multi-threaded TTY program, adding a time tag in front of the $ sign of each POS message. A typical example of this is given by:

```
2693843$PASHR,POS,1,08,072241.00,5832.73619,N,01458.14602,E,+00121.03,,288,001,+000,01,00,01,00,1H00*60
2694854$PASHR,POS,1,08,072242.00,5832.73636,N,01458.14531,E,+00121.13,,291,001,+000,01,00,01,00,1H00*64
2695846$PASHR,POS,1,08,072243.00,5832.73651,N,01458.14464,E,+00121.05,,291,001,+000,01,00,01,00,1H00*62
```

The depth data is given with three decimal accuracy and separated by a > character. Even though it is not visible in the example below, the different depth readings are also separated by a Carriage Return. A typical depth data example is given by:

```
2696357>+05.0952696457>+05.0812696557>+05.0852696657>+05.0802696757>+05.0792696857>+05.0772696958>+05.0802697048>+05.0802697148>+05.0772697248>+05.090
```
5 Simulations

This chapter describes some of the MATLAB particle filter simulations made in order to give a qualitative evaluation of the algorithm performance. Different simulations have been made both on simulated data from a map provided by FOI in Stockholm and on experimental data acquired during the depth soundings described in Chapter 4. Although no actual MATLAB code will be presented, this chapter will start with a description of some specific details of the implementation used in these simulations. This is then followed by a description of some simulations made on the FOI map and, finally, a description of some simulations made on the SBUS map.

5.1 Simulation Model

As described in Chapter 3, the conceptual TNS model relies on surrounding navigation systems for receiving information about momentary course and velocity. The simulations described in this chapter are all based on that model, but the origin of the course and velocity data is somewhat different between the simulations made on simulated data and on experimental data.

5.1.1 Evaluation Tracks

In order to simulate the behaviour of the positioning algorithm, a series of corresponding measurements regarding course, velocity and depth must be available. Naturally, the true position data must also be provided in order to evaluate the positioning accuracy. These series of corresponding data are referred to as evaluation tracks or, when no risk of ambiguity exists, just tracks.

As described in Chapter 4, a few evaluation tracks were made during the depth soundings. These are not part of the data set used to create the sea chart, and can therefore give a good knowledge about the particle filter navigation performance. The momentary course and velocity from the experimental data has been calculated off-line, based on the sequential movement between consecutive GPS positions.

When it comes to the simulations based on the FOI map, no predefined evaluation tracks exist. Instead, simulated evaluation tracks are constructed based on a desired starting position, course and velocity. Using this information, the resulting positions for each track point is calculated. These positions, and their corresponding map depths, are then considered to be the true positions and depths respectively. In order to model the difference between map depth values and measured depth values, a simulated measurement noise is added. In the same manner, simulated measurement noise is added to the originally desired course and velocity.
5.1.2 The State Equation

For both simulated and experimental data, the track points are naturally related to the state transition equation (2.3a). The track data is given at discrete time indices \( t \) separated by the sample time \( T \), which should be the same sample time used to propagate the particle cloud. As described in Chapter 2 and 3, the \( u_t \) vector in the state transition equation describes the true movement, based on course and velocity. When it comes to movement calculated from GPS positions, an additive noise \( v_t \), modelling the movement uncertainty, would probably give the most accurate description. However, additive noise does not reflect the most probable nature of the conceptual TNS model movement uncertainty. Therefore, a more accurate state equation, modelling the simulations made both on the FOI map and the SBUS map, is given by

\[
x_{t+1} = f_t(x_t, u_t, v_t).
\]

In this case, \( v_t \) is two-dimensional and describes the modelled, independent, additive course and velocity noise respectively.

The actual implementation of (5.1) used in these simulations is given by the following two-dimensional equation

\[
x_{t+1} = \begin{pmatrix} y_{t+1} \\ x_{t+1} \end{pmatrix} = \begin{pmatrix} y_t + T \cdot (v(t) + v_v) \cdot \sin(a(t) + v_a) \\ x_t + T \cdot (v(t) + v_v) \cdot \cos(a(t) + v_a) \end{pmatrix} \leftrightarrow \text{East Pos.} \\
\begin{pmatrix} y_{t+1} \\ x_{t+1} \end{pmatrix} = \begin{pmatrix} y_t + T \cdot (v(t) + v_v) \cdot \sin(a(t) + v_a) \\ x_t + T \cdot (v(t) + v_v) \cdot \cos(a(t) + v_a) \end{pmatrix} \leftrightarrow \text{North Pos.}
\]

Here, there might be a risk for ambiguity if the reader does not observe that the same variable names (\( x, y, v \) and \( x, y, v \)) are used for different purposes in different contexts. The two-dimensional state variable \( x_t \) consists of the east position co-ordinate \( y_t \) and the north position co-ordinate \( x_t \). Note that these co-ordinates can be distinguished from the state variable \( x_t \) and the measured depth \( y_t \) by the use of non-italic and italic characters respectively. The two-dimensional noise variable \( v_t \) consists of the course process noise \( v_v \) and the velocity process noise \( v_a \). By once again noticing the use of italic characters, the process noise should not be misinterpreted as the velocity \( v(t) \). Finally, the variable \( a(t) \) is the compass course, starting with zero degrees to the north and counting increasing values clockwise. Thus, the north axis \( x \), the east axis \( y \) and the downward axis \( z \) define a positive right-hand system. These definitions follow the Swedish geodetic reference frame standards set by the Swedish land-surveying agency, Lantmäteriet.

5.1.3 Process Noise and Measurement Noise

Naturally, all measurements from any system suffer from some measurement noise. This would also be the case when measuring course, velocity, travelling depth and distance to the seabed (normally referred to as measured depth). Hopefully, the nature of these measurement noise components would be relatively well known for any given AUV. In these simulations, however, the measurement noise must be partially simulated.

54
During the SBUS map simulations, the nature of the depth measurement noise is for instance illustrated by Figure 5.11 in Section 5.3. It originates from the differences between the generated SBUS map and the evaluation track depths. Even though it does not have a true Gaussian nature, it is nevertheless approximated with Gaussian noise when making the probability estimation during the measurement update. Although the approximation might seem a little rough, it works relatively well.

During the FOI map simulations, a depth measurement noise, with a nature similar to the actual experienced noise from the SBUS map simulations, was added to the created evaluation tracks. Some fictitious measurement noise was also added to both course and velocity data. In all simulations, the process noise \( v_t \) was chosen large enough to sufficiently cover the (modelled) maximum possible course and velocity measurement noise. The exact values of these noise parameters are all accounted for in the following sections, describing each specific simulation example.

5.2 Evaluation on Simulated Data

This section describes some particle filter simulations based on simulated data from the FOI map. Before presenting the details of these simulations, the general nature and origin of the FOI map is described.

5.2.1 The FOI Map

In earlier studies on terrain navigation, e.g. [And 00], a more or less accurate terrain map of an unspecified area of Swedish territorial waters was used for positioning simulations. Thanks to Ingemar Nygren at FOI and Anders Hallgren at Metria Geo SE, this sea chart has been made available for simulations here. This sea chart will from now on be referred to as the FOI map (Figure 5.1).

Highly accurate sea charts of Swedish territorial water are generally surrounded with severe restrictions for security reasons. The knowledge of which areas are covered and what accuracy has been achieved is naturally not information that can be made available for master’s thesis studies. However, highly accurate sea charts with resolutions as high as one by one or two by two meter’s do exist, and an educated guess would tell that their numbers are increasing as results of ongoing mapping projects [And 00].

To the best of our knowledge, the FOI map has the typical characteristics of the average broken ground within Swedish territorial waters. However, in order not to reveal classified information, the FOI map has been re-scaled and distorted before being made available for public studies. Since the true nature of the FOI map is not entirely specified, the results from these simulations should be seen mainly as qualitative indications of general algorithm behaviour.
Since the exact nature of the FOI map is not known anyway, both resolution and depth has been re-scaled to imitate the depth interval of the SBUS map. The chosen area of the FOI map has a size of 100 by 500 grid points. During the simulations on this map, the map resolution is said to consist of one by one meter’s squares. Therefore, the navigation results will be presented in terms of meter positioning errors. Note that all such numbers are based on the assumption that the nature of the FOI map, when assigned this resolution, still reflects the parameters of actual underwater terrain. True or false, these kind of assumptions can easily be assessed afterwards, if such information could be made available for further evaluation. Still, the one by one meter’s grid assumption does not in any way cripple the results since the navigation accuracy quite easily can be reinterpreted in terms of other grid resolutions.

Figure 5.1  The FOI map describes an unspecified area of Swedish territorial waters. Before being made public, the map has been distorted and re-scaled in order not to reveal restricted information. However, an educated guess would tell that it still has typical seabed characteristics.
5.2.2 Simulation I

All of the simulations I, II and III are made within the same area of the FOI map (Figure 5.2). The fictitious measurement noise is also the same between these three simulations, as well as the number of particles, as seen in Table 5.1 and Figure 5.3.

<table>
<thead>
<tr>
<th>Number of Particles</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Depth Measurement Noise (Gaussian)</td>
<td></td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>$\sigma = 0.05$ m</td>
</tr>
<tr>
<td>Actual Depth Measurement Noise (Sinus Shaped)</td>
<td></td>
</tr>
<tr>
<td>Amplitude</td>
<td>0.05 m</td>
</tr>
<tr>
<td>Frequency</td>
<td>3.3 Hz</td>
</tr>
<tr>
<td>Assumed Depth Measurement Noise (Gaussian)</td>
<td></td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>$\sigma = 0.08$ m</td>
</tr>
</tbody>
</table>

*Table 5.1. Parameters for Simulation I, II and III.*

Simulation I is made for the purpose of illustrating the robustness of the particle filter algorithm against course and velocity measurement noise with zero bias. As seen in Table 5.2 and Figure 5.4, the amplitude of the added measurement noise is relatively large compared to the actual course and velocity values. Still, the positioning results have an accuracy within one meter throughout the entire simulation (besides during the initial stage). The result from Simulation I is illustrates by Figure 5.5 and Figure 5.9.

| Actual Process Noise (Experienced Disturbance) |
| Course  | $\sigma = 2^\circ$ (Gaussian) |
| Velocity| $\sigma = 0.03$ m/s (Gaussian) |
| Assumed Process Noise |
| Course  | $\sigma = 5^\circ$ (Gaussian) |
| Velocity| $\sigma = 0.05$ m/s (Gaussian) |

*Table 5.2. Parameters for Simulation I.*

**Figure 5.2**
The 100 by 500 meter's area of the FOI-map and the evaluation-track used for Simulation I, II and III.
Figure 5.3  The measurement noise added to the map depth is the same the three simulations I, II, and III. As seen above, the depth ranges between nine and fourteen meters in this 500-samples track. The used noise has a Gaussian component with a 5 cm standard deviation added on top of a 5 cm sinus shaped oscillation of 3.3 Hz. (Note: Depth is plotted against the track samples, not against travelled distance.)

Figure 5.4  Simulation I. Course and velocity with measurement noise plotted sample by sample. For both course and velocity, the added noise is Gaussian and unbiased.

Figure 5.5  Absolute values of the positioning error from Simulation I plotted sample by sample. The short initial phase, before reaching accuracy below one meter, is a strong indication of the particle filters ability to handle noisy signals, as these above, if the terrain contains essential information.
5.2.3 Simulation II

During simulations II and III, the amplitude of the added course and velocity measurement noise was chosen much smaller than in Simulation I. Instead, a +2 degree course bias and a +2% velocity bias was added (Table 5.3 and Figure 5.6). The nature of the process noise used for modelling the measurement noise was not changed from Simulation I, though. Thus, these two simulations can be seen as an illustration of the robustness of the particle filter algorithm against unknown bias-errors, provided that the terrain contains sufficient variation.

As seen in Figure 5.7 and Figure 5.9, the position estimate is immediately affected by the course bias and keeps straying away from the true track more or less constantly during the first 200 meters (or 275 iterations). After this phase, however, the terrain

<table>
<thead>
<tr>
<th>Actual Process Noise (Experienced Disturbance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course</td>
</tr>
<tr>
<td>Course Bias</td>
</tr>
<tr>
<td>Velocity</td>
</tr>
<tr>
<td>Velocity Bias</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assumed Process Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course</td>
</tr>
<tr>
<td>Velocity</td>
</tr>
</tbody>
</table>

Table 5.3. Parameters for Simulation II and III.

Figure 5.6 Course and velocity measurement noise from Simulation II and III plotted by sample. The course noise is Gaussian with 0.05 cm standard deviation on top of 2° bias. The velocity noise is also Gaussian but with 0.01 m/s standard deviation on top of 2% bias.

Figure 5.7 Absolute value of positioning error and the total non-normalised particle weights from Simulation II. The low total particle weight around sample 275 clearly indicates particle filter divergence.
starts to provide enough information for the particle filter to slowly begin to converge back towards the true track.

Also in Figure 5.7, the total non-normalised particle weight\(^7\) for Simulation II is given. It is defined according to (2.29) with a particle weight (importance weight) for each particle given by

\[
    w^i = e^{-|e^i/\alpha|^2} \quad \text{for} \quad i = 1, ..., N. \tag{5.3}
\]

Hence a zero-residual corresponds to a particle weight of unit one, which makes the maximum total non-normalised particle weight the same as the number of particles \(N\).

The maximum positioning error during this simulation (besides the initial phase) is about five meters, and occurs after approximately 200 meters. The corresponding dead-reckoning error at this moment (275 iterations) would have been around nine meters. As mentioned before, at this moment the particle filter starts to converge towards the true track and ends up with less than one meter positioning error at the end of the simulation. This should be compared to the corresponding dead-reckoning error, which would have ended at around sixteen meters.

### 5.2.4 Simulation III

As illustrated by Figure 5.7, a particle filter divergence can be detected in Simulation II. Around iteration 275, a number of approximately ten consecutive iterations resulted in a very low value of the total non-normalised particle probability weight. As mentioned in Section 2.3.6, this could be used to restart the particle filter, providing the current estimated position as the new user provided starting-location. Such a restart is illustrated by Simulation III.

![Figure 5.8](image.png)

**Figure 5.8** Absolute value of positioning error and the total un-normalised particle weights from Simulation III. Due to the low total particle weight around sample 275 in Simulation II, the effects of a filter restart with the old position estimate as new starting position was tested in Simulation III.

\(^7\) The total un-normalised particle weight is in Chapter 2 referred to as the total un-normalised importance weight.
As mentioned in the previous section, all parameters are the same for both Simulations II and III. The positioning error for Simulation III is illustrated in Figure 5.8 and Figure 5.9. After the initial phase, the position estimate stays around one meter from the true position. Even though the positioning error is roughly the same size for both Simulation II and III at the end of the simulations, the results from Simulation III illustrates that there could be a lot to gain from a filter restart, if divergence is detected. The immediate three-meter error decrease shortly after restart is naturally highly desirable. The restart can also be seen as an attempt to eliminate the risk for total particle filter position degeneration. This is needed, since there is no guarantee that the terrain in general will contain enough information for an automatic filter convergence as in Simulation II.

Figure 5.9  Particle filter position estimate from Simulation I (left), Simulation II (middle) and Simulation III (right). Simulation I has relatively large but unbiased course and velocity measurement noise. Simulation II and III suffer from course measurement noise with +2° bias and velocity measurement noise with +2% bias, as shown in Figure 5.6. As shown in Figure 5.7, a filter divergence can be detected in simulation II after 275 iterations. A particle filter restart, starting at the estimated position at that time, is simulated by Simulation III.
5.3 Evaluation on Experimental Data

Besides the simulations made on the FOI map, a series of simulations on the experimental data from the depth soundings has also been made. One general impression from these simulations is that the FOI map contains much more terrain variations than the SBUS map. This is e.g. reflected by the relatively short cloud size reduction time during the initial phase of the FOI map simulations. In the SBUS map simulations, the duration of this phase is significantly longer.

Although the SBUS map contains less terrain information, it is still possible to receive accurate position estimates from the particle filter during these simulations. Two simulations on the SBUS map will be accounted for here. One of those (Simulation IV) produces highly accurate results while the other (Simulation V) produces an interesting positioning degeneration, which will be referred more in detail.

Both of these simulations will be made on the same evaluation track, under the same conditions and with roughly the same particle filter parameters. The chosen track is the diagonal south-east to north-west oriented track, seen in Figure 5.10. This track starts at about nine meters depth, which slowly shallows to eight meters, then back to nine again.

![Figure 5.10](image_url)  
*Figure 5.10 The SBUS map and the three evaluation-tracks derived during the depth soundings. The evaluation track used in Simulation IV and V is the diagonal southeast to northwest oriented track.*
After that, an approximate 45° slope changes the depth to about sixteen meters, which then slowly levels out to about seventeen meters (Figure 5.11). The entire length of this evaluation track is about 490 meters, measured at an average velocity of 1.4 m/s.

One main benefit of using experimental data for algorithm evaluation is the unknown nature of the imperfections of the available data set. As seen in Figure 5.11, the measured evaluation track depths differ quite a bit from the generated terrain map. A visual inspection easily tells that the difference, the residual, is neither Gaussian, independent or unbiased. The assumption about ideal, Gaussian, zero-mean depth measurement noise, used when assigning weights to the particles, still poses no bigger problem. To include the bias and most of the residual peaks, the standard deviation of the presumably Gaussian residual-distribution is merely chosen a bit larger than what had been needed for an ideal residual. The most severe impacts of this choice are merely slightly slower particle filter convergence and initial cloud size reduction phase.

*Figure 5.11* Above, the depth profile of the evaluation track used in Simulation IV and V is plotted together with the corresponding map depth. Below, the difference between the track depth and map depth is plotted with the map depth as zero reference.
Figure 5.12 Characteristics of the evaluation track used in Simulation IV and V. All variables are plotted iteration by iteration, i.e., with a sample time of one second. The eastwards and northwards momentary movement in meters is given by dy and dx respectively. The corresponding velocity in m/s is given by v. Finally, the compass course is given by α, and the measured depth in meters by d.

5.3.1 Simulation IV

As mentioned earlier, a number of qualitative simulations with varying results were made on the SBUS map. Simulation IV is one example of what can be achieved at best, with a good choice of particle filter parameter. The particle filters in Simulation IV and V are both initiated with 2500 particles, scattered around the user provided initial position.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Particles</td>
<td>2500</td>
</tr>
<tr>
<td>Initial Position Uncertainty</td>
<td>(Gaussian)</td>
</tr>
<tr>
<td>$\sigma_x = 18 \text{ m}$</td>
<td>$\sigma_y = 18 \text{ m}$</td>
</tr>
<tr>
<td>Assumed Depth Standard Deviation</td>
<td>(Gaussian)</td>
</tr>
<tr>
<td>$\sigma = 0.15 \text{ m}$</td>
<td></td>
</tr>
<tr>
<td>Assumed Process-Noise Course</td>
<td>$\alpha = 5^\circ$ (Gaussian)</td>
</tr>
<tr>
<td>Velocity</td>
<td>$\sigma = 0.3 \text{ m/s}$ (Gaussian)</td>
</tr>
</tbody>
</table>

Table 5.4 Parameters used in Simulation IV. No additional simulated course or velocity measurement noise has been added.
In both cases, the user provided starting position is given by the true position. The signals used by the particle filter are visualised in Figure 5.12, where the velocity $v$ (m/s) and the compass course $a$ (degrees) is the parametric input and $d$ (m) is the measured depth.

Table 5.4 contains most of the particle filter parameters. Note the relatively high standard deviation values of the assumed course and velocity process noise. The 0.3 m/s velocity measurement-error is quite large compared to the 1.4 m/s average velocity. The 5° angular spread during each propagation is also rather high. Such large values would make the size of the area covered by the particle cloud to expand rather rapidly, if the terrain provided little information. It would also give each particle a large degree of freedom to propagate astray from the movement vector given by the parametric input. Depending on the illusiveness of the terrain, such freedom could both increase and decrease particle filter performance, as will be illustrated by Simulation V. In this case however, these large values might be seen as an indication of the robustness of the particle filter, provided an appropriate value of the assumed depth standard deviation for the given terrain.

![Figure 5.12](image)

Figure 5.13  Particle filter results from Simulation IV, plotted iteration by iteration. The positioning error for y (East), x (North) and the absolute positioning error is given by the three upper plots (units in meters). Below, the total non-normalised particle weight is given. In this framework, the maximum value of this sum is the same as the number of particles. This is a natural result of the choice to assign a zero-residual an non-normalised particle weight of unit one, as described by (5.3).
The particle filter results from Simulation IV is given by Figure 5.13 and Figure 5.15. After the initial cloud size reduction phase (See Figure 5.14), the algorithm receives a high amount of terrain information at the 45° slope. This occurs around sample 125, and is indicated by the dip in total non-normalised particle weight in Figure 5.13. This is the natural performance of a more or less spread-out particle cloud when propagated into an information-rich terrain, rendering many particles highly unlikely. After this event, the positioning error stays within one meter throughout the rest of the simulation. Despite the large process noise and the relative flatness of the terrain from that point and on, the cloud size does not expand much more than which is the case at that moment, as illustrated by Figure 5.15.

### 5.3.2 Simulation V

The parameters used in Simulation V are generally the same as in Simulation IV (Table 5.5). The only difference is the slightly larger value of the assumed depth standard deviation. This minor change, in combination with the nature of the chosen track and terrain, will however result in an interesting particle filter performance. The main reason to include Simulation V here is therefore not the accuracy of the navigation results, but rather its pedagogic qualities.

The interesting difference between the two simulations occurs immediately after the 45° slope around sample 125, as illustrated by Figure 5.15. There, the position estimate of Simulation V makes a turn eastwards for about 60 iterations and then returns. The main reason for this behaviour

<table>
<thead>
<tr>
<th>Number of Particles</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Position Uncertainty (Gaussian)</td>
<td>( \sigma_x = 12 ) m ( \sigma_y = 12 ) m</td>
</tr>
<tr>
<td>Assumed Depth Standard Deviation</td>
<td>( \sigma = 0.17 ) m (Gaussian)</td>
</tr>
<tr>
<td>Assumed Process Noise</td>
<td>Course ( \sigma = 5^\circ ) (Gaussian)</td>
</tr>
<tr>
<td>Velocity ( \sigma = 0.3 ) m/s (Gaussian)</td>
<td></td>
</tr>
</tbody>
</table>

*Table 5.5 Parameters used in Simulation V. No additional simulated course or velocity measurement noise has been added.*
is the large difference between the measured depth and the depth according to the terrain map, as illustrated earlier in Figure 5.11.

One of the basic characteristics of the particle filter is that its momentary performance logically depends on the entire history of iterations. As a result of this, the main origin of this behaviour difference is found prior to the 45° slope, i.e. more or less in the cloud size reduction phase. As it turned out, many outlier particles were not terminated during earlier iterations in the same extent as in Simulation IV. Thus, the slightly larger assumed depth standard deviation in Simulation V caused a slower cloud size reduction phase and thus also a more sway-wise spread out cloud, as seen in Figure 5.16. When the measured depth started to differ significantly from the map values, the map depth of the outlier particles happened to coincide better with the measured depth than those particles located around the true position. Therefore, the weight centre of the particle cloud was shifted towards the particles in the upper right corner.

Figure 5.15 The different behaviour of Simulation IV (left) and Simulation V (right). Both simulations suffer from the same large residuals close to iteration 125, but their different particle cloud shapes result in different performance.
Figure 5.16 As a result of the lower value of the assumed depth standard deviation $\sigma$ in Simulation IV, the particle cloud is less scattered in that case (left) than in Simulation V (right) at iteration 125. The larger sway-wise spread in Simulation V makes the algorithm more vulnerable to large residuals.

\[ \sigma = 0.15 \, \text{m} \]
\[ \sigma = 0.18 \, \text{m} \]

Figure 5.17 Particle filter results from Simulation V, plotted iteration by iteration. The figures illustrate the positioning error for y (East), x (North), the absolute positioning error and the total non-normalised particle weight. The positioning errors are all given in meters.
Figure 5.18 illustrates the whole outcome of the depth discrepancy in Simulation V. Here it can be seen that there still remained some particles close to the true position, despite the false position estimate. As soon as the residual grew smaller, the outliers became less probable than the particles close to the true position, and were consequently terminated.

Finally, a remark about the overall positioning results of Simulation V is in place. As seen in both Figure 5.15 and Figure 5.17, the positioning results in the flat section of the map are not as good as in Simulation IV. After iteration 180, the mean positioning error remains in the range of five to ten meters. This happens in spite of the fact that the particle cloud constantly either touches or slightly covers the true position. This should be interpreted as a result of relatively poor available terrain information in combination with an overly residual-tolerant choice of assumed depth standard deviation. As the parameters are set, the particles close to the true position do not gain enough particle weight for the cloud to converge on a position centred on the true position. In other words, with a particle filter set-up like this, the terrain does not provide information enough to give better accuracy. However, since this kind of positioning error does not grow by the hour, a positioning accuracy as given in Simulation V may still be accurate enough for a terrain navigation system.
Figure 5.18  The particle clouds in Simulation V after 152, 160, 170 and 180 iterations respectively. As seen by the figures, the particle filter almost diverges around iteration 160 due to the difference between the measured depth and the evaluation track depth (see Figure 5.11). Some particles still remain close enough to the true position and soon gain some probability weight. At the same time, the particles representing false positions lose weight. At iteration 180, almost all of the outlier particles in the upper right corner have been terminated.
6 Results

When initially studying the terrain navigation problem, a number of different approaches to terrain navigation were generally reviewed. When studying the literature available on the topic, the many benefits of using a particle filter for this kind of problem soon became apparent. This way, one of the major aims was met, giving a theoretical description and a mathematical solution to the terrain navigation problem, using the Bayesian approach.

Along the way, a conceptual Terrain Navigation System model was defined. This TNS positioning model is relatively general and is based on a fixed single narrow-beam sonar. The model illustrates how to take advantage of the carrier vehicle’s already existing sensors to a rather high extent. Based on a MATLAB implementation of this model, a number of qualitative simulations have been made. The major aim of these simulations has been to give qualitative indications to the behaviour and performance of a particle filter under varying conditions, and an illustrative selection from these simulations has been accounted for here.

Within the boundaries of this work, a depth chart over a small area of the Swedish lake, Lake Vättern has been created. The chosen area was roughly covered with parallel measurement tracks more or less separated by five meters. The final resolution of the resulting depth chart was chosen as one by one meters.

In addition to the created depth chart, a sea chart over a vaguely specified terrain area of Swedish territorial waters was examined. This created an opportunity to make MATLAB simulations on underwater terrain with different characteristics. The main differences between these depth charts have been generally described. However, due to the relatively poor knowledge about the true nature of the borrowed sea chart, at this stage no unconditional conclusions should be made from these simulations. Although, when used for qualitative simulations only, the results from this sea chart are still highly interesting, as they give indications to what is achievable with information-rich terrain.

6.1 Conclusions

One of the most confident conclusions of this work is that the terrain navigation problem is technically solvable. Even though a number of details still remain to be investigated further, no grave obstructions have been discovered. The most crucial obstacle to overcome, before an experimental TNS could be made operational, is the access to high-resolution sea charts. Without sufficiently detailed charts over varying terrain, the use of terrain navigation is obviously not feasible. This problem is, however, more related to security restrictions and willingness to co-operate and share information between agencies than to technical issues.
Still there remains a large amount of investigations concerning sensor choice, algorithm implementation, real-time performance etc. After that follows the issues surrounding the integration of a TNS into a vehicle autopilot. Such matters take time, and have their natural place in any future design and construction phase of an experimental TNS. The profound and irrefutable conclusion, however, is that the mathematical and conceptual issues of terrain aided underwater navigation are manageable. The MATLAB simulations also strongly indicate that terrain navigation, even when applied to relatively flat terrain, can provide a highly accurate and vital increase in a vehicle’s INS position estimate.

6.2 Future Work

The recommendations and suggestions for future work can be divided into the following four main categories.

6.2.1 System Performance

How high is the computational burden of an embedded implementation of the particle filter? This factor will be crucial for the real-time on-line performance of a future TNS, and should, therefore, be investigated thoroughly.

- The computational burden of the particle filter should be estimated in order to determine the system requirements of a future embedded on-line application. This would be simplified if the particle filter code was first implemented in i.e. C or any similar programming language.

- The detection of particle filter divergence, based on the sum of non-normalised particle weights, and the impact of automatic particle filter restarts should be investigated further.

- The possibility of dynamically varying the number of particles as well as re-sampling less often (i.e. using the Sampling Importance Sampling) should be investigated. What impact could such actions have on the computational burden under real-time performance condition demanding a deterministic position-update rate?

6.2.2 Theoretical Aspects

Some theoretical aspects of the TNS concept remain to be examined.

- The possibility of using the information given by the Cramér-Rao lower bound for identifying terrain suitable for navigation purposes should be investigated more thoroughly.
• How would the introduction of another state variable, such as vehicle course, impact on the accuracy of the position estimate? How much would it increase the computational burden?

• How could a travelling depth bias error be handled? Would the introduction of an extra state variable solve the problem, or would it be better handled outside the particle filter algorithm, perhaps by an EKF?

6.2.3 Sensor Choice
Which would be the best sensor to use for continuous depth measurements? How should accuracy, equipment size and energy consumption or increased danger of detection from amplified signature due to use of active sensors be considered?

• Altimeters/Narrow-beam sonars. Could the sonar frequency be chosen in a domain with high levels of acoustic background noise, hiding the signal with e.g. use of signal specific filters, i.e. using whispering sonars?

• Lasers. Would it be possible to use lasers instead of acoustic sensors, thus eliminating the extra increase in acoustic signature?

• Doppler logs. A Doppler log used for velocity measurements already provides some sort of depth measurement. How could the geometrical pattern of the four Doppler log sonar-beams be used to derive even more information compared to four independent depth measurements?

6.2.4 Simulations
An obvious continuation would be to make further simulations on the derived data material. Some quantitative simulations, such as Monte Carlo simulations, would make a significant complement to the qualitative simulations made so far. A number of Monte Carlo simulations could easily be made for varying conditions, such as certain terrain variation, map accuracy, measurement noise and system noise etc. This way, a better knowledge achievable average positioning accuracy of the particle filters would be gained.
References and Bibliography


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnt som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Tobias Karlsson